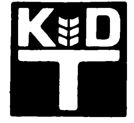


Speicherprogrammierbare Steuerungstechnik

Lehrbrief 2

Logikverarbeitung mit Mikrorechner

KDT-Fernkurs



KDT-Fernkurs
Speicherprogrammierbare Steuerungstechnik

Lehrbrief 2
Logikverarbeitung mit Mikrorechner

3., unveränderte Aiföage

Dr. K. Mauersberger
Dr. D. Fischer

Kammer der Technik
Präsidium
Fachverband Elektrotechik
Sekretariatsbereich Weiterbildung
Berlin 1987

Logikverarbeitung mit Mikrorechner : Lehrbrief / Mauersberger, Klaus ; Fischer, Dieter. - Berlin : Präsidium d. Kammer d. Technik, 1987. - 52 S.- (KDT-Fernkurs Speicherprogrammierbare Steuerungstechnik ; 2)

Clara-Zetkin-Str. 115/117, Berlin, 1086

I 12 4 Ag 238/66/87

Printed in the German Democratic Republic

Redaktionsschluß: 30. Januar 1983

Inhaltsverzeichnis Seite

5. Stand und Trend der programmierbaren Steuerungstechnik

5.1. Analyse angebotener Steuerungseinrichtungen

Aus der Literatur und aus Firmenunterlagen wurden 60 speicherprogrammierbare Steuereinrichtungen von ≈30 international bekannten.Herstellern analysiert. In den folgenden Abbildungen sind die gebrauchswertbestimmenden Kennwerte gegenübergestellt. In der Abb. 1 sind für 3 Bereiche Aussagen zusammengefaßt. In den vorderen 5 Spalten stehen Werte, die die zentrale Verarbeitungseinheit charakterisieren.

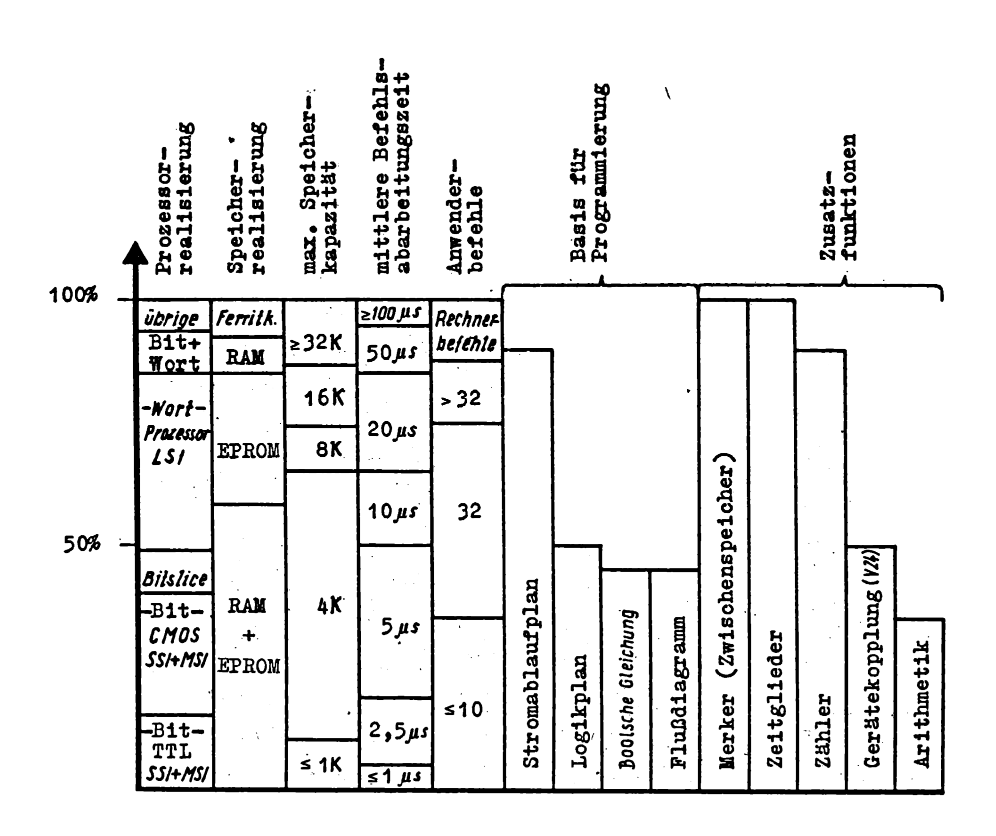


Abb. 1 Charakteristik speicherprogrammierbarer Steuerungen (zentraler Verarbeitungsteil)

Als Prozessor werden speziell aufgebaute Schaltungen zur Bitverarbeitung in 40 % der PC eingesetzt. Den niedrig- (SSI) und mittelintegrierten Schaltkreisen (MSI) in komplementärer MOS-Technologie wird aus Gründen der Störsicherheit und niedrigem Leistungsverbrauch der Vorrang eingeräumt. Reichlich 30 % der PC's sind mit universellen Mikroprozessoren ausgerüstet.
Diese wortorganisierten Elemente können außer der direkten Logikverarbeitung eine Vielzahl weiterer Aufgaben (Organisation, Adreßrechnung, Eigendiagnose, Arithmetik) bei Vorhandensein entsprechender Programme übernehmen. Der Einsatz von universellen Prozessoren bringt aber auch Nachteile (Initialisierung nach jedem Zuschalten der Versorgungsspannung, längere Abarbeitungszeit für eine Logikanweisung des Anwenders), so daß in anspruchsvollen Steuerungen für komplexe Prozesse gegenwärtig mit Kombinationen aus speziellem Bit- und universellem Wortprozessor gearbeitet wird. Eine Sonderlösung stellen die Verarbeitungseinheiten aus extrem schnellen, mikroprogrammierbaren Bitsliceprozessoren dar. Ein Element (Scheibe = slice) besitzt eine Verarbeitungsbreite von 2 oder 4 bit und kann meist bis 32 bit kaskadiert werden. Diese Elemente werden nur von wenigen Bauelementeproduzenten angeboten und sind nur mit Unterstützung von Rechnern effektiv programmierbar.

Die Abb. 1 zeigt weiterhin die prozentuale Verteilung zur Speicherrealisierung und maximal zulässigen Speichergröße. Im Normalfall werden bekanntlich die Programme

- Anwenderprogramm: Anweisungsfolge des Anwenders zur Bildung der gewünschten Logikverknüpfung

- Betriebsprogramme: Programmteile des Steuerungsherstellers zum Laden, Anzeigen und Abarbeiten von Anwenderprogrammen und zur Initialisierung und Überwachung der Funktionsfähigkeit der Steuerausrüstung

in einem resistenten Speicher (ROM, EPROM) abgelegt, während für variable Daten Schreib-Lese-Speicher genutzt werden. Die extrem verlustleistungsarme CMOS-Technologie bietet aber auch die Möglichkeit, Programme durch ständige Batteriepufferung im RAM einzuschreiben.

Der erforderliche Speicherraum für Anwenderprogramme ist nicht

nur von der Zahl der Logikgleichungen und der durchschnittlichen Menge von Verknüpfungen je Gleichung abhängig, sondern auch von der Effektivität der Befehle.

Für eine einfache Notierung der Anwenderlogik ist ein ausgewogenes Verhältnis zwischen stärkerer Spezialisierung der Befehle und der Übersichtlichkeit und Handhabbarkeit der Befehlsliste herzustellen. 30 % der PC bieten dem Anwender 16....20 problemorientierte Befehle. Weniger als 12 % der untersuchten Steuerungen lassen nur die ursprünglichen Mikrorechnerbefehle zur Logikprogrammierung zu. Im Abschnitt 6 wird gezeigt, welche Probleme dabei entstehen, so daß diese Möglichkeit nur als Notlösung angesehen werden kann.

Im 2. Bereich ist die Form der Logikbeschreibung angegeben, die als günstigste Grundlage für die Programmierung anzusehen ist. Alle Beschreibungsformen sind mit mehr oder weniger Aufwand ineinander überführbar. Aus der prozentualen Verteilung ist zu ersehen, daß ein großer Teil der Steuerungen mehrere Eingabemöglichkeiten zuläßt. Der Stromlaufplan (Kontaktplan der äquivalenten Relaisschaltung) wird von amerikanischen Herstellern bevorzugt.

Die Bedeutung von Zeit- und Zählvorgängen in der industriellen Steuerungstechnik zeigt die 3. Gruppe der Spezial- oder Zusatzfunktionen. Auch die Möglichkeit, Zwischenvariable abzuspeichern, bieten alle Hersteller an.

Bezüglich der Realisierung von Arithmetikoperationen werden Logiksteuerungen ständig erweitert. Hier ist ein direkter Zusammenhang zwischen Einsatz von Wortprozessoren und Arithmetik gegeben. Gleichzeitig werden gegenwärtig numerische Steuerungen in Richtung Logikverarbeitung erweitert, so daß es in naher Zukunft keine klare Aufgabentrennung zwischen numerischen und nichtnumerischen Steuereinrichtungen geben wird.

Wichtige Klassifizierungsmerkmale für Prozeßein- und -ausgangskarten enthält die Abb. 2. Man erkennt, daß bei mehr als 75 % der angebotenen Einrichtungen eine Potentialtrennung angeboten wird. Diese Funktion wird meist über Optokoppler realisiert. Teilweise werden aus Preisgründen auch Kartenbaugruppen mit Relais für diesen Zweck angeboten.

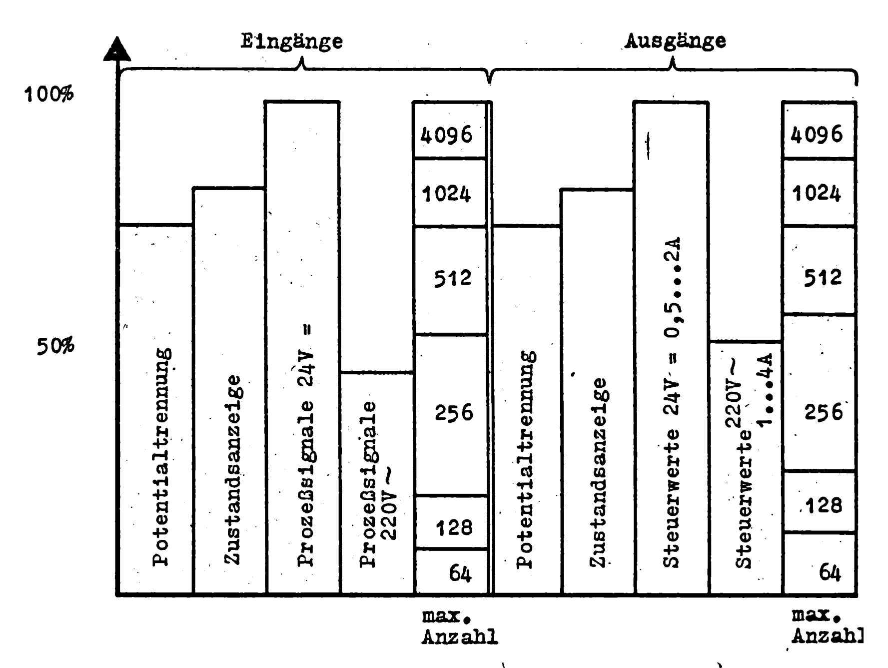


Abb. 2 Charakteristik speicherprogrammierbarer Steuerungen (Ein- und Ausgangskarten)

Bei der Zustandsanzeige, die generell mit Lumineszensdioden ausgerüstet ist, liegt der Prozentsatz über 80 %. Auf weitere Eigenschaften, die eine Erhöhung der Verfügbarkeit bewirken, wird in der Abb. 2 nicht eingegangen. Viele Ein- und Ausgangskarten weisen zusätzlichen Schaltungsaufwand auf, der eine Selbstüberwachung der Funktionsfähigkeit oder eine Prüfbarkeit durch die zentrale Verarbeitungseinheit ermöglicht.

Für Prozeßein- und -ausgänge haben sich bei Gleichspannungsbetrieb 24 V als standardisierte Spannung durchgesetzt. Rund die Hälfte der Steuerungen können über entsprechende Karten auch direkt mit 220 V Wechselspannung gekoppelt werden. Für die Ausgänge wird eine strommäßige Belastung bis 4 A angeboten, so daß die errechneten Steuerwerte direkt Stellhandlungen bewirken können. Die maximale Zahl der Prozeßanschlüsse an Steuerungen

wird durch die Adreßwortbreite, die Zahl der Steckplätze und die Zahl der Anschlüsse je Kartenbaugruppe beeinflußt. Aus der Abb. 2 ist als typischer Wert die Zahl 256 zu entnehmen. Die Zahl ist für alle Einzelmaschinensteuerungen ausreichend. Für komplexe Automatisierungsanlagen (Kraftwerke usw.) sind mehr Signale zu verarbeiten. In diesen großflächigen Anlagen werden teilweise dezentralisierte Steuerungen projektiert, die hierarchisch gekoppelt werden, um Kabelaufwand zu sparen.

5.2. Vor- und Nachteile von PC mit universellem Wortprozessor

Universelle Prozessoren und insbesondere Mikroprozessoren ermöglichen den Aufbau von PC's mit minimalem Schaltungsaufwand (Hardware) durch den Einsatz von hoch- und höchstintegrierten Schaltungen (LSI, VLSI),

Die Bauelementehersteller bieten für ihre Prozessoren vollständige Familien von Ergänzungsschaltkreisen (RAM, PROM, Bustreiber, Zeitgeber, parallele und serielle Ein- und Ausgabeelemente) an, die ohne zusätzlichen Logikaufwand zusammengeschaltet werden können. Durch die weiter oben erwähnten Zusatzfunktionen lassen sich informationelle Kopplungen zu installierten Prozeßrechnern oder von PC's untereinander einfach realisieren. Damit ist der Übergang von der Automatisierung einzelner Aggregate zur koordinierten Steuerung von Maschinensystemen und ganzen Produktlinien gerätetechnisch einfacher möglich.

Durch die vom PC-Hersteller erstellten Betriebsprogramme ist für universelle Prozessoren eine Anpassung an unterschiedliche Eingabesprachen (verschiedene Befehlslisten bzw. verschiedene Mnemoniks eines Befehlsspektrums) erreichbar.

Weitreichende Möglichkeiten ergeben sich bei Wortprozessoren zur Ansteuerung peripherer Geräte (Lochbandleser, -stanzer, Tastaturen, Bildschirmeinheiten und Drucker), die wahlweise zur Eingabe, Anzeige, Änderung und syntaktischen Prüfung von Anwenderprogrammen genutzt werden können. Diese Elemente werden auch zur Fehlersuche in Programmen und zur Statussignalisation bei Abweichungen vom Regelbetrieb genutzt. Damit erreichen PC mit einem Wortprozessor als Zentraleinheit eine wesentlich größere Flexibilität. Sie bieten dem Nutzer auch viel mehr Komfort.

Nachteilig beim Einsatz von Rechnerkernen als PC ist der größere Zeitbedarf zur Befehlsabarbeitung, so daß größere Zykluszeiten für die Summe aller Steuergleichungen entstehen und für schnelle Prozesse u.U. unzulässige Totzeiten entstehen. Weiterhin weisen diese Steuerungen bei sehr einfachen Einsatzfällen preisliche Nachteile auf, da die entwickelten Softwarepakete gegenwärtig hohe Kosten bedingen. Obwohl die Programmierung und der normale Betrieb für den Anwender bei Rechnersteuerungen einfach ist, sind für Service und Reparatur speziell geschulte Personen erforderlich. Ein Eingriff des Anwenders in die Betriebsprogramme zur nachträglichen Anpassung der Betriebsweise ist kaum erfolgversprechend.

6. Beschreibung ausgewählter Befehle und Befehlsgruppen des U 880

Der Mikroprozessorschaltkreis U 880 ist ein leistungsfähiger, mit hoher Geschwindigkeit arbeitender, universeller LSI-Schaltkreis, der als Zentraleinheit in Rechnern und Steuerungen (Steuerrechner) zum Einsatz kommt.

Neben einem umfangreichen Befehlssatz besitzt er eine Reihe - hardwaremäßig realisierter - Baugruppen, die seine Anwendung vereinfachen und damit einen effektiven Einsatz ermöglichen.

Durch die 8 bit parallele Verarbeitung ist er besonders für arithmetische Aufgaben geeignet.

Neben dem Akkumulator besitzt er eine Reihe weiterer Register (B, C, D, E, L, H...), die als Notizblockspeicher mit schnellem Zugriff genutzt werden können.

Selbstverständlich besteht auch die Möglichkeit, zwischen externen Speicherbaugruppen und Ein- und Ausgangseinheiten Datentransporte vorzunehmen.

Den peripheren Einheiten werden vom Anwender Adressen zugeordnet. Der Aufruf erfolgt über separate Adreßleitungen des Prozessors (Adreßbus). Die Daten werden über die Datensammelleitungen (Datenbus) übertragen. Für die Koordination und Organisation des Ablaufes sind die Steuersignale (Steuerbus) erforderlich.
Eine Blockdarstellung des U 880 zeigt Abb. 3.

Von der Vielfalt der Befehle sollen im folgenden wesentliche Gruppen näher beschrieben werden. (Durch Modifikation der 158 Befehle können 692 verschiedene Instruktionen gebildet werden.)

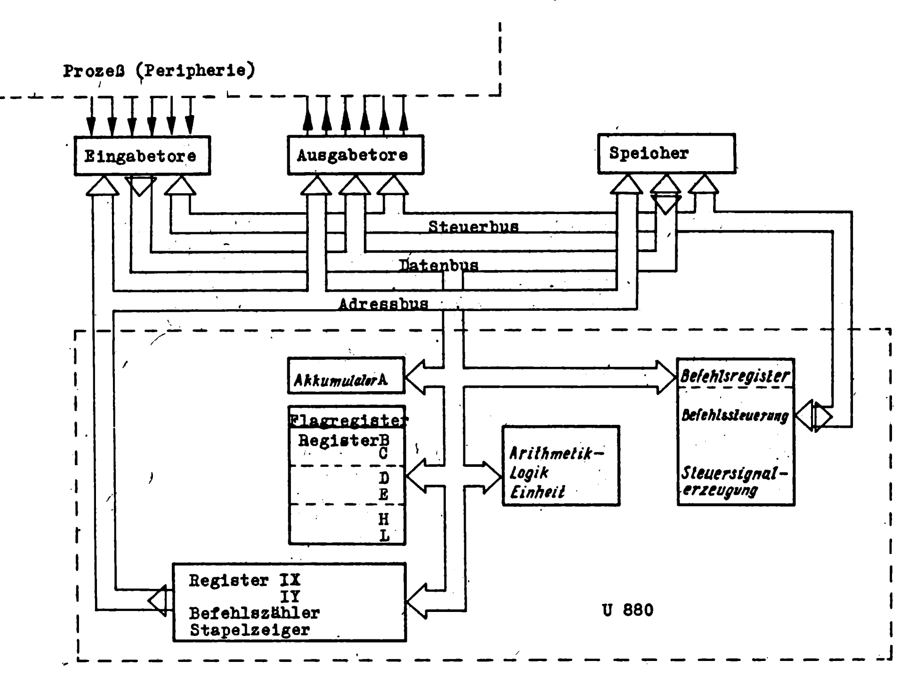


Abb. 3 Blockschaltbild des U 880 (vereinfacht)

Ladebefehle

Die Ladebefehle realisieren den Datentransport zwischen den verschiedenen inneren und äußeren Speicherplätzen. Ihre Abarbeitungszeit ist unterschiedlich lang, wobei Transporte innerhalb der Notizblockspeicher am schnellsten vollzogen werden, während Datentransporte zu den peripheren Speichern bei indirekter Adressierung wesentlich länger dauern. Für Echtzeitverarbeitung sollten also möglichst die internen Register genutzt werden. Im allgemeinen werden durch die Ladebefehle die Flags nicht beeinflußt. Dies gestattet zwischen Operationen, bei denen die Flags beeinflußt und ausgewertet werden, Ladebefehle einzufügen.

Blocktransferbefehle

Mit diesen Befehlen wird der Transport von Datenblöcken (aufeinander folgende Adressen) elegant gelöst. Der einzelne Ladebefehl wird nach Modifikation interner Register wiederholt, bis eine Abbruchbedingung erfüllt ist. So wird durch den Befehl LDIR

(DE) : = (HL) ausgeführt, danach wird
DE : = DE + 1
HL = HL + 1
BC : = BC - 1

Mit der symbolischen Schreibweise (DE) : = (HL) wird ausgedrückt, daß der Inhalt der Speicherzelle, die durch HL (16 bit) adressiert wird, in die Speicherzelle eingetragen wird, deren Adresse im Register DE (16 bit) steht.

Der Befehl wird wiederholt, bis BC = 0.

Bei evtl. auftretenden Interupts während der Abarbeitung dieses Befehls muß beachtet werden, daß die Register DE, HL und BC in die Statusrettung einzubeziehen sind.

Logikbefehle

Zu dieser Befehlsgruppe gehören die Befehle UND, ODER, logische Negation und Antivalenz (Exklusives ODER).

Ein Operand steht stets in A, während der zweite Operand in den U 880 Registern bzw. im Speicher abgelegt sein kann.

Die Operationen werden für jedes bit parallel ausgeführt und das Ergebnis wieder in A eingeschrieben.

Bei den Logikbefehlen werden das Z-(Null) und S-Flag (Vorzeichen) beeinflußt. Das S-Flag wird gesetzt, wenn das Ergebnis der Verknüpfung der höchstwertigen Bits gleich Eins ist. Das Z-Flag wird gleich Eins, wenn nach der Verknüpfung der Akkumulatorinhalt gleich Null ist.

Für Steuerungsaufgaben werden die logischen Verknüpfungen meist für 1 bit (z.B. D0) ausgeführt. Es ist dabei zu beachten, daß andere Bitstellen immer einen definierten Zustand einnehmen, damit keine ungewollte Flagbeeinflussung eintritt, denn für das Ergebnis kann neben der signifikanten Bitstelle (D0) auch das Z-Flag genutzt werden.

Soll das Z-Flag zur weiteren Auswertung des Ergebnisses einer logischen Operation herangezogen werden, so muß man sicherstellen, daß außer der signifikanten Ergebnisstelle die verbleibenden Bits zwangsweise Null sind (Maskierung). Weiterhin ist zu beachten, daß das Z-Flag immer den negierten Wert der logischen Operation repräsentiert.

Bei Belegung der Bitstelle D7 mit den Werten der zu verknüpfenden Variablen signalisiert das S-Flag das wahre Ergebnis, und die Maskierung kann entfallen.

Bitmanipulationsbefehle

Mit diesen Befehlen können einzelne Bits im Datenwort beeinflußt werden. Bei den Testbefehlen wird stets der negierte Wert des ausgewählten Bits im Z-Flag abgebildet. Eine Weiterverarbeitung bzw. Auswertung dieser Bitstelle mit bedingten Sprüngen bietet sich an.

Die SET- bzw. RES-Befehle gestatten das Beeinflussen eines Bits im Datenwort mit 0 bzw. 1. Diese Befehlsgruppe wird für das Setzen der Steuerwerte gern genutzt.

Sprungbefehle

Die Sprungbefehle gestatten, die Abarbeitungsfolge der Befehle zu beeinflussen. Es können Programmteile übersprungen werden, wenn bestimmte Bedingungen erfüllt sind. Die Sprungbefehle gestatten auch den Aufbau zyklischer Programme. Die Sprungbedingung bezieht sich immer auf den Inhalt eines Flags.

Für Logikverarbeitung nutzt man diese Befehle oft, wenn mit Bit-Testbefehlen gearbeitet wird. Bei der Abarbeitung von Sprungbefehlen werden die Flags nicht beeinflußt.

Ein-Ausgabe Befehle

Diese Befehle realisieren den Verkehr mit den peripheren Einheiten und Baugruppen (max. 256). Die Adresse der peripheren Einheit kann direkt im Befehl notiert sein oder sich im C-Register befinden. Für die Ein- und Ausgabe von Datenblöcken existieren Blocktransferbefehle.

7. Abarbeitung logischer Funktionen mit Mikrorechner

7.1. Direkte Programmierung

Unter direkter Programmierung wollen wir verstehen, daß der Anwender für die Umsetzung der logischen Gleichungen unmittelbar den Befehlssatz des Mikrorechners nutzt.

Diese Programmierebene erfordert vom Programmierer eine genaue Kenntnis des genutzten Rechners und viel Aufmerksamkeit, wobei die Fehlerhäufigkeit der Programme selbst bei versierten Bearbeitern hoch ist.

Der entscheidende Vorteil dieser Methode besteht in der Möglichkeit, Programme mit minimaler Abarbeitungszeit zu erstellen. Der gravierendste Nachteil der direkten Programmierung liegt im Verlust der ursprünglichen Logikstruktur. Dies hat zur Folge, daß eine Inbetriebnahme, Fehlersuche und nachträgliche Änderung solcher Programme schwierig ist. Außerdem entstehen für ein System logischer Gleichungen bei der unabhängigen Programmierung durch mehrere Bearbeiter ganz unterschiedliche Befehlsfolgen, deren Effektivität stark durch die Fertigkeiten und Erfahrungen des Programmierers beeinflußt werden. Gut "lesbar" bleiben die Befehlsfolgen nur für den Urheber. Die allgemeine Transparenz ist gering. Der hohe manuelle Aufwand der direkten Programmierung begrenzt die Anwendung auf kleine und mittlere Logikprobleme und auf Einsatzfälle, die keine Umprogrammierung in kurzen Zeitabständen erfordert, so daß ein Großteil der prinzipiellen Flexibilität verloren geht. Das Prinzip konnte sich in industriell hergestellten PC nicht durchsetzen, findet aber bei individuellen Rationalisierungsmaßnahmen, die Rechner verwenden, Anwendung.

7.1.1. Nutzung der Logikbefehle des Mikrorechners

Für die ersten Überlegungen soll davon ausgegangen werden, daß in einem Programmteil vor der Logikverarbeitung die Variablen von digitalen Eingabekarten durch Inputbefehle (IN) byteweise in einen vorgegebenen Speicherbereich des RAM transportiert wurden. Ebenso byteweise werden die vom Rechner generierten Steuerwerte durch Outputbefehle (OUT) an Ausgabekarten transferiert und dort gespeichert.

Für die sequentielle Abarbeitung der Logik werden außer Ladebefehlen, die Operationen AND, OR und CPL (Komplement) benutzt. Da, wie schon erwähnt, diese Logikbefehle auf das gesamte Rechnerwort gleichermaßen wirken, muß durch zusätzlich eingeschobene Verschiebe- bzw. Rotationsbefehle das zu verarbeitende (signifikante) Bit des adressierten Wortes an eine fest vereinbarte Stelle (im Beispiel DO) verschoben werden. Bei nur einer Schieberichtung und angenommener Gleichverteilung für die Häufigkeit der Variablen in allen 8 Bitpositionen ergeben sich vor jeder Logikverknüpfung im Durchschnitt 4 Verschiebebefehle. Bei Nutzung beider Schieberichtungen halbiert sich diese Zahl.

Ein weiteres Problem dieser Verarbeitungsmethode ist, daß für die Logikbefehle nicht alle Adressierungsarten des Prozessors nutzbar sind und die Logikverknüpfung nur im Akkumulator durchführbar ist.

Für das Beispiel

$$Y127 = X014 \; X022 \vee \overline{\overline{X036} \; X020}$$

sei folgende Bedeutung der Indizes vereinbart:

- 1. und 2. Stelle (Hexaziffern) stellen die Kanaladresse und gleichzeitig den niederwertigen Adreßteil des Speicherbereiches im RAM-Speicher dar.
  (der höherwertige Adreßteil ist einmalig festgelegt - im Programmbeispiel 10 H)
- 3. Stelle gibt die Bitposition im Wort an (0....7).

Zeile	Assemblermnemonik		Bemerkung
1	LD	A, (1001 H)	
2	RRCA		
3	RRCA		
4	RRCA		
5	RRCA		
6	LD	B, A	
7	LD	A, (1002 H)	
8	RRCA		
9	RRCA		
10	AND	B	
11	LD	C, A	
12	LD	A, (1003 H)	

```
13      RLCA
14      RLCA
15      CPL
16      LD      B, A
17      LD      A, (1002 H)
18      AND     B
19      CPL
20      OR      C
21      AND     01 H            Maske 1
22      LD      B, A
23      LD      A,(1012 H)
24      RLCA
25      AND     0FEH            Maske 2
26      OR      B
27      RRCA
28      LD      (1012 H), A
```

Programm I

Das entstandene Programm I weist bis zur Zeile 20 keine Besonderheiten auf. Für die Ladeoperationen wird die direkte Adressierung benutzt, da sie für diesen Fall den geringsten Speicherbedarf bei kleinster Abarbeitungszeit garantiert. Mit der Maske 1 werden die störenden Stellen D1....D7 des generierten Ergebnisbytes ausgeblendet und im Register B zwischengespeichert. Nachfolgend wird zunächst das Wort gelesen, das den Wert des vorangegangenen Zyklus enthält. Über die Verschiebung und die Maske 2 wird Y127 auf Null gebracht und durch die ODER-Operation mit dem aktuellen Wert belegt. Die Verschiebungen lassen sich an dieser Stelle durch geänderte Programmstruktur reduzieren. Es ergibt sich allerdings der Nachteil, daß die Maske 2 in diesem Fall der Bitposition der Ergebnisstelle angepaßt werden muß. Das entstandene Programm ist nicht optimal für den U 880. Die benutzten Anweisungen sind allerdings auch für die Abarbeitung mittels U 808 grundsätzlich geeignet.

Für den U 880 kann die Folge ab Zeile 22 wie angegeben geändert werden:

```
        LD.     HL,   1012 H
        JPZ     M1
        SET     7, (HL)
        JP      M2
   M1   RES     7, (HL)
   M2
```

Durch die Verwendung der Bitmanipulationsbefehle wird das Programm effektiver.

Der Nachteil der großen Zahl von Verschiebebefehlen läßt sich auf 2 Wegen beseitigen. Setzt man byteorientierte Ein- und Ausgabekarten voraus, dann ist es zeitsparend, wenn nach jedem IN-Befehl (Akkumulator enthält 8 aufeinanderfolgende Variable) 8 Speicherschreiboperationen mit zwischengeschobenem Verschieben des Akkus und Inkrementieren von HL ausgeführt werden. HL wurde vor Beginn mit der Speicheranfangsadresse des X-Vektors geladen. Der Zyklus wird für jeden Eingabekanal wiederholt. Durch dieses kurze Programmstück wird jede Variable auf einen Speicherplatz des RAM an einer festen Bitposition separiert. Die nichtsignifikanten Bitstellen sind in ihrer Belegung unbestimmt. Da die erzeugten Ausgangsvariablen auch jeweils eine feste Bitposition einer adressierbaren RAM-Zelle belegen und die freien Bits identisch "0" sind, kann durch eine Folge aus ODER- und Verschiebebefehlen ein Wort zusammengefügt werden. Dieser Akkumulatorinhalt gelangt über OUT zur Peripherie. Die angegebene Parallel-Serien- und Serien-Parallel-Wandlung erfordert einen erhöhten Speicherplatzbedarf. Durch die wesentlich einfacheren Programme und eine verkürzte CPU-Zeit wird der Nachteil garantiert kompensiert.

Im Abschnitt 7.2. wird auf dieses Prinzip zurückgegriffen. Eine ganz andere Möglichkeit ergibt sich, wenn bitorientierte E/A-Karten eines Bitprozessors eingesetzt werden (LB1). Durch die niederwertigste Hexade der Adresse kann über Multiplexer auf den Eingangskarten die Prallel-Serien-Wandlung ohne Programmaufwand realisiert werden. Die Rücktransformation übernehmen die Demultiplexer der Ausgangskarten. Für die Gleichung aus dem 1. Beispiel

$$Y47 = X14 \; X22 \vee \overline{\overline{X36} \; X20}$$

kann unter diesen Voraussetzungen das Programm II geschrieben werden.

Zeile	Assemblermnemonik		Bemerkung
1	IN	14	
2	LD	B, A	
3	IN	22	
4	AND	B	
5	LD	C, A	
6	IN	36	
7	CPL		
8	LD	B, A	
9	IN	20	
10	AND	B	
11	CPL		
12	OR	C	
13	OUT	47	

Programm II

Man erkennt sofort, daß das Programm wesentlich einfacher geworden ist. Eine unmittelbare Umsetzung beliebiger logischer Gleichungen in Befehlsfolgen gelingt bei Anwendung von Logikbefehlen nicht immer. Bei konjunktiven oder disjunktiven Normalformen stößt man nie auf Schwierigkeiten und auch bei einfachen Klammerausdrücken oder Termnegationen (wie das Beispiel zeigt) ist die Lösung naheliegend. Doch selbst in dem angegebenen Beispiel bringt die entsprechende Umformung in die disjunktive Normalform ein kürzeres Programm.

7.1.2. Nutzung der Test- und Sprungbefehle

Schon im Lehrbrief 1 wurde darauf verwiesen, daß logische Operationen mit bedingten Sprüngen ausführbar sind. Durch die Möglichkeit, im Befehlssatz des U 880 einzelne Bits auszutesten, wird eine elegante Programmierung ermöglicht.

Für die Programmerstellung eignet sich hier der Programmablaufgraph oder ein Flußdiagramm besser als die logischen Gleichungen (Abb. 4).

Negationen sind leicht ausführbar, indem man die Sprungbedingung umkehrt bzw. bei Termnegationen das Sprungziel ändert. An dem bekannten Beispiel soll dies näher erläutert werden.

$$Y127 = X014\ X022 \lor \overline{\overline{X036}\ X020}$$

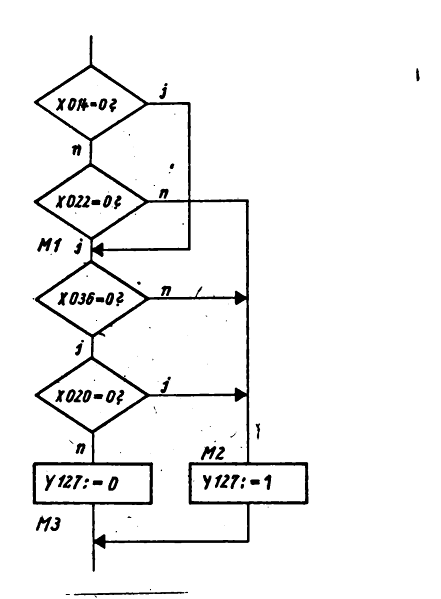


Abb. 4 Programmablaufgraph für Programm III

Das Programm III kann folgende Form erhalten:

Zeile		Assemblermnemonik		Bemerkung
1		Bit	4, (IX + 01)	
2		JRZ	6	Sprungziel M1
3		BIT	2, (IX + 02)	
4		JRNZ	18	Sprungziel M2
5	M1	BIT	6, (IX + 03)	
6		JRNZ	12	Sprungziel M2
7		BIT	0, (IX + 02)	
8		JRZ	6	Sprungziel M2
9		RES	7, (IX + 12)	
10		JR	4	Sprungziel M3
11	M2	SET	7, (IX + 12)	
12	M3			

Ein Vorteil der Sprungstruktur zur Abarbeitung logischer Gleichungen liegt in der Systematik des Programmaufbaus. Am Beispiel sieht man, daß für die Verknüpfungsseite je Variable genau 1 Bittest- und 1 Sprungbefehl benötigt werden. Weiterhin sind im Programm III nur relative Sprünge verwendet worden. Diese

belegen nur 2 Byte im Programm und ermöglichen eine Sprungweite von ± 127 Byte mit absoluter Zahlenangabe im Befehl. Durch diese Eigenschaft bleibt auch der Maschinenkode des Programmes unabhängig von den belegten Speicheradressen. Die Zahl der Variablen in einer Gleichung wird allerdings auf maximal 21 begrenzt, da pro Variable durch die 2 benötigten Befehle 6 Byte belegt werden. Die indexierte Adressierung eignet sich bei dieser Programmstruktur besonders gut. Das Indexregister IX des Prozessors ist vor der Logikverarbeitung mit der Anfangsadresse des Feldes von Ein-, Ausgangs- und Zwischenvariablen geladen worden. Für Normalformen von logischen Ausdrücken (ohne Klammer und Termnegation) kann ein einfacher Algorithmus zur Festlegung der Sprungbedingung und der Sprungweite angegeben werden (Abb. 5).

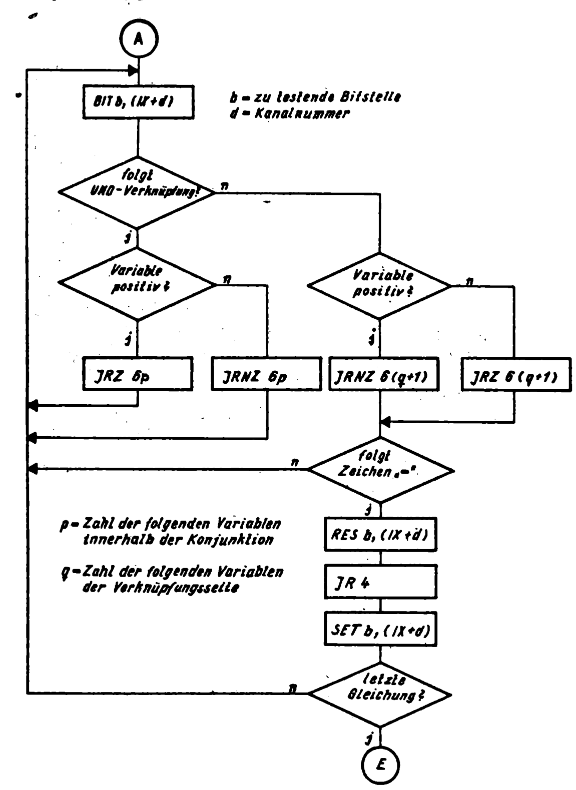


Abb. 5 PAP zur Erstellung eines Programmes mit Sprungstruktur

Die Variablen sind in einem RAM-Speicherbereich dicht gepackt. Durch die Programmierung treten die Variablenadressen in den Testbefehlen direkt wieder auf. Die folgenden Sprungbefehle überspringen unwesentliche Variable, wodurch die Abarbeitungszeit nicht konstant ist und nur als Mittelwert angegeben werden kann. Die Zeiten sind etwa gleich der Variante II. Der Speicherplatzbedarf liegt zwischen Variante I und II. Sehr vorteilhaft ist aber, daß praktisch beliebige Gleichungsformen bzw. Ablaufdiagramme programmiert werden können.

Einschränkungen ergeben sich nur durch die begrenzte Sprungweite von + 129. Bei disjunktiven Normalformen können in einer Gleichung maximal 21 Variable enthalten sein, da für jede Variable die Sprungweite 6 beträgt.

Sehr schwierig ist die Einordnung der Tabellenverfahren, Die Abspeicherung der vollständigen Wertetabelle ist durch die starke Begrenzung der Variablenzahl ohne Bedeutung als industrielle Steuerung. Alle anderen Verfahren benötigen außer der logikspezifischen Tabelle noch mindestens einen logikunabhängigen Programmteil zur Abarbeitung der Tabelle in Verbindung mit den aktuellen Eingangsbelegungen. Werden vom Anwender beide Teile programmiert, so ergeben sich nur Vorteile, wenn die Logik oft geändert werden muß, da nur die direkte Tabelle neu aufgestellt werden muß. Da sich aber bezüglich Zeitbedarf und Speicherkapazität für praktisch interessante Variablenzahlen größere Werte ergeben als in den vorgestellten Varianten, wird dennoch kein Gebrauch davon gemacht.

	Programm		
	I	II	III
max. Variablenzahl	256 . 8	256	129 . 8
Speicherplatzbedarf des Verarbeitungsprogrammes je Variable (Byte)	≈ 8	≈ 3,5	≈ 7
max. Variablenzahl pro Gleichung	256 . 8	256	21
Zeitbedarf f. Verarbeitung einer Variable (µs)	≈ 14	≈ 7	≈ 7
Speicherplatzbedarf für 1 Variable im RAM	1 Bit	0	1 Bit
max. Klammertiefe	3	3	unbegrenzt

Abb. 6 Zusammenstellung einiger Parameter der Programmiervarianten

7.2. Problemorientierte Programmierung

Die meisten Anwender und Nutzer von Steuerungen interessiert die interne Arbeitsweise der Einrichtung nicht. Diese wollen sie als "black box" betrachten, da sie nur Mittel zum Zweck ist. Von der Steuerung sind für den Käufer außer Preis, Beschaffungsmöglichkeit, Größe und Zuverlässigkeit vor allem die steuerungstechnische Leistungsfähigkeit an den Anschlüssen und der Aufwand zur Anpassung an die jeweilige Steuerungsaufgabe wichtig. Durch die problemneutral produzierten PC muß der Anwender sich zwangsweise mit der Programmierung befassen. Die Hersteller kommen dem Nutzer durch problemangepaßte Fachsprachen entgegen. Für die Herausbildung der speziellen Fachsprachen der Steuerungstechnik (in der Literatur, z.B. /2/, auch Steuersprachen genannt) wurde die Symbolik und das Vokabular der Beschreibungsmittel der konventionellen Steuerungstechnik als Basis verwendet. In den PC mit speziellem Bitprozessor ist der Satz der Instruktionen unmittelbar auf die Hardware abgestimmt, so daß die primäre Anwendernotierung ohne vorherige Umformung (Übersetzung) im Speicher der PC abgelegt wird und auch in dieser Form von der zentralen Logikeinheit abgearbeitet wird. Bei Einsatz von universellen Mikroprozessoren erfolgt der Programmlauf nur auf der Ebene der Maschinensprache. Die prozessorfremde Anwendernotierung muß entweder einmalig vor Inbetriebnahme in die Maschinensprache übersetzt werden (compilierende Arbeitsweise) oder während der Logikverarbeitung wird jede Instruktion des Anwenders durch eine Folge von Maschinenbefehlen ausgeführt (interpretierende Arbeitsweise).

Für die Darstellung des Verhaltens von Steuerungen gibt es bekanntlich mehrere gleichberechtigte Beschreibungsmittel. Dem entsprechend existieren auch unterschiedliche problemangepaßte Notierungen als Ausgangspunkt zur Programmierung. Wie im Lehrbrief 1 schon erwähnt, kennt man 2 grundsätzliche Verfahren der Beschreibung von Automaten. Zu den zustandsorientierten Möglichkeiten gehört beispielsweise der Automatengraph und der Programmablaufgraph. Obwohl für einfache nicht parallelläufige Prozesse übersichtliche Notierungen bei geeigneter Mnemonik entstehen, wird diese Methode bei den bekannten industriellen PC nicht verwendet. Die Hersteller bieten aber häufig sogar mehrere strukturorientierte Notierungsformen an. Die Struktur

einer elektronischen Steuerung ist beispielsweise durch einen äquivalenten Stromlaufplan mit Relais darstellbar. Diese Methode ist bei Herstellern aus den USA weit verbreitet. Der wichtigste Vorteil entsteht für die Anwender bei der Ablösung herkömmlicher Kontaktsteuerungen. Für die Programmierung ist keine neue Symbolik zu erlernen. Das Eingabegerät besitzt eine funktionsorientierte Tastaturzuordnung mit den Mindestelementen aus Abb. 7.

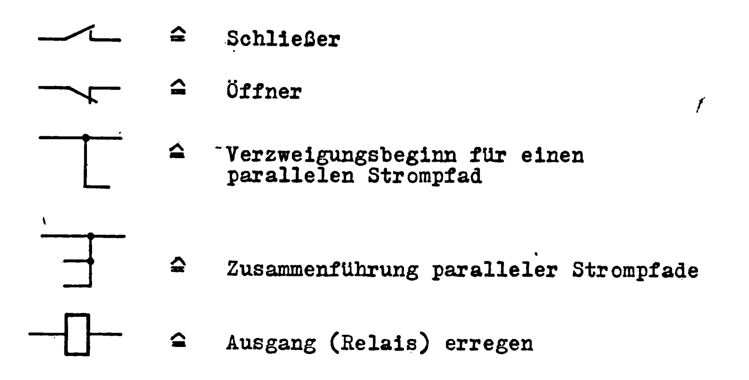


Abb. 7 Mindestsymbolmenge für Kontaktplandarstellung

Für die Numerierung der Ein- und Ausgänge wird zusätzlich eine Zehnertastatur eingesetzt. Über weitere Symbole sind Zähl- und Zeitglieder implementierbar. Für das Kontrollesen des eingegebenen Programmes ist ein graphisches Display am besten geeignet. Für das Programmiergerät ergibt sich dadurch ein hoher Grundaufwand in der Hardware und im Monitorprogrammsystem. In Abb. 8 ist als einfaches Beispiel eine Wendeschützschaltung mit direkter Umsteuerung gezeigt. Die horizontale Darstellung ergibt sich durch die Bildschirmorganisation. Die Platzierung und u.U. die Änderung von Symbolen kann der Bediener über einen Kursor beeinflussen. Die Kapazität des Displays ermöglicht im Durchschnitt bei 5 parallelen Zweigen max. je 7 Kontakte.

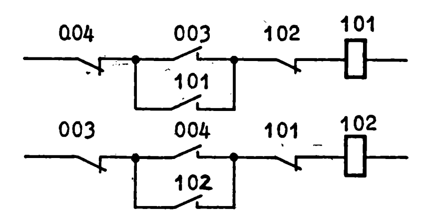
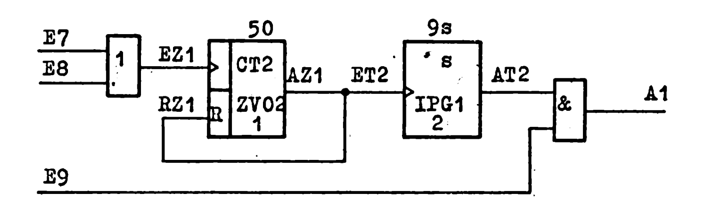


Abb. 8 Wendeschützsteuerung als graphischer Kontaktplan

Bei größerem Umfang muß der Programmierer Zwischenvariable bilden. Ein beachtenswerter Aspekt dieser Programmiervariante ist, daß die orginalen Relaisstromlaufpläne durch Restriktionen der ursprünglich eingesetzten Bauelemente große Redudanzen enthalten können. Es entsteht u.U. ein wesentlich längeres, umständlicheres und langsames Programm als aufgabengemäß bedingt. Nur bei billigen Speichermedien und sehr schnellen Prozessoren entstehen keine zeitlichen Nachteile beim Einsatz.

Eine äquivalente graphische Methode ist auch für Logiksymbole denkbar, aber bisher in der Steuerungspraxis nicht üblich. Es sind aber Sprachen bekannt, die bausteinorientiert angelegt sind (DOLOG 80, PROLOG 2). Für alle üblichen logischen Symbole, die auch hardwaremäßig als niedrig- oder mittelintegrierte Schaltkreise in verdrahteten Steuerungen verwendet werden, wurde eine Mnemonik vereinbart. Vom Anwender brauchen damit z.B. Adder, Zähler, Zeitglieder oder Register nicht mehr explizit beschrieben werden. Diese komplexen Funktionen sind genau so wie UND, ODER, NAND durch ihren Namen definiert. Der Anwender muß in verbaler Form nur die allgemeinen Bausteineingänge mit konkreten Signalen der Steuerung (Ausgänge vorgeschalteter Elemente) belegen. Die Form der Programmierung bewirkt bei sehr komplexen Steuerungsaufgaben eine vorteilhafte Verkürzung des Quellprogrammes. Das einfache Beispiel aus Abb. 9 zeigt das Prinzip, kann aber das Vermögen nur andeuten.

In der 1. Spalte der Mnemonik steht der Name des Bausteines. Sind mehrere Modifikationen eines solchen Elementes möglich (bei Vorwärtszähler z.B. binär oder dezimal), so wird dies



Zeile	Mnemonik		Bemerkung
n	ODRB	E7, E8, EZ1	letzte Stelle Ausgang
n + 1	UMBI	AZ1, RZ1	Umspeicherbefehl
n + 2	ZV02	1, 50	Zählernummer, Bereich
n + 3	UMBI	AZ1, ET2	Umspeicherbefehl
n + 4	IPG1	2, 90, DS	Nummer, Dauer, Dezisek.
n + 5	UNDB	AT2, E9, A1	letzte Stelle Ausgang
			IPG: Impulsgeber

Abb. 9

durch eine Zahl im Namen ausgedrückt. Eine Sonderstellung nimmt der Umspeicherbefehl ein, der einen bereits definierten Ausgang einem Eingang zuordnet und damit die Leitungsverbindung repräsentiert. In der 2. Spalte sind bei einfachen kombinatorischen Elementen die zu verknüpfenden Signale angegeben. Das letzte Signal stellt vereinbarungsgemäß jeweils den Ausgang dar. Bei komplexen Bausteinen können Zählbereich, Zeiten (Maßzahl und Maßeinheit) u.ä. entsprechend der Programmiervorschrift aufgelistet werden. Detaillierte Angaben findet man in der Programmieranleitung zur Steuerung ursalog 5020 /3/.

Der entscheidende Nachteil dieser effektiven Form der Quellnotierung liegt in dem hohen Programmaufwand zur Umsetzung in ein lauffähiges Maschinenprogramm des verwendeten Rechners. Es kommt deshalb hier nur die compilierende Arbeitsweise in Frage. Das Übersetzerprogramm des Programmiergerätes benötigt zusammen mit den übrigen Bedien- und Testroutinen 8...32 K Speicherworte. Bei diesem Aufwand kommt der Preis des Programmiergerätes in die Größenordnung der eigentlichen Steuerung.

Als 3. Methode hat sich besonders für mittlere Systeme die Notierung der logischen Gleichungen eingeführt. Auch hier wird von jedem Steuerungshersteller eine spezielle Mnemonik angeboten. Obwohl sich international eine Standardmenge an zulässigen Anweisungen gebildet hat, werden die Sprachen weiter auf Betriebsebene entworfen und durch mehrere Einflußfaktoren modifiziert. Als Operanden werden mindestens Eingangs-, Ausgangs- und Zwischenvariablen zugelassen, Als Operatoren dürfen neben UND, ODER, NEG vielfach auch Klammer und u.U. Antivalenz, bedingter Sprung und Arithmetik geschrieben werden. Die zugelassenen Zeichen sind meist eine Untermenge der Schreibmaschinentastatur, so daß übliche Datenendgeräte zur Notierung des Quellprogrammes eingesetzt werden können.

Als Beispiel sei die Quellnotierung für die PC 600 /4/ vom VEB Numerik "Karl Marx" zum Schaltbeispiel aus Abb. 8 gegeben.

N 10: A 101 = / E 4 . (E 3 + A 101) . / A 102
N 11: A 102 = / E 3 . (E 4 + A 102) . / A 101

In einer Quellzelle wird stets eine logische Gleichung (Satz) notiert. Jede Zeile beginnt mit N als Symbol für die Satznummer. Nach einem Doppelpunkt folgt die Ergebnisseite der Gleichung und nach dem "=" die Verknüpfungsseite. Andere Varianten der Notierung logischer Gleichungen wurden im Lehrbrief 1 bereits gezeigt.

7.2.1. Compilierende Arbeitsweise

Für jede der bisher angegebenen problemorientierten Quellnotierungen kann durch ein Übersetzerprogramm die Überführung in ein lauffähiges Maschinenprogramm realisiert werden. In Abb. 10 ist für das Verfahren ein grober PAP zu sehen.

Es ist üblich, daß der Anwender vor dem eigentlichen Logikteil Angaben zur Belegung der Steckplätze der Steuerung mit Eingabe- bzw. Ausgabekarten, den Zeitwerten der Zeitglieder des Programmes und u.U. des Speicherbereiches für das Anwenderprogramm dem Übersetzer mitteilt (Vereinbarungsteil).

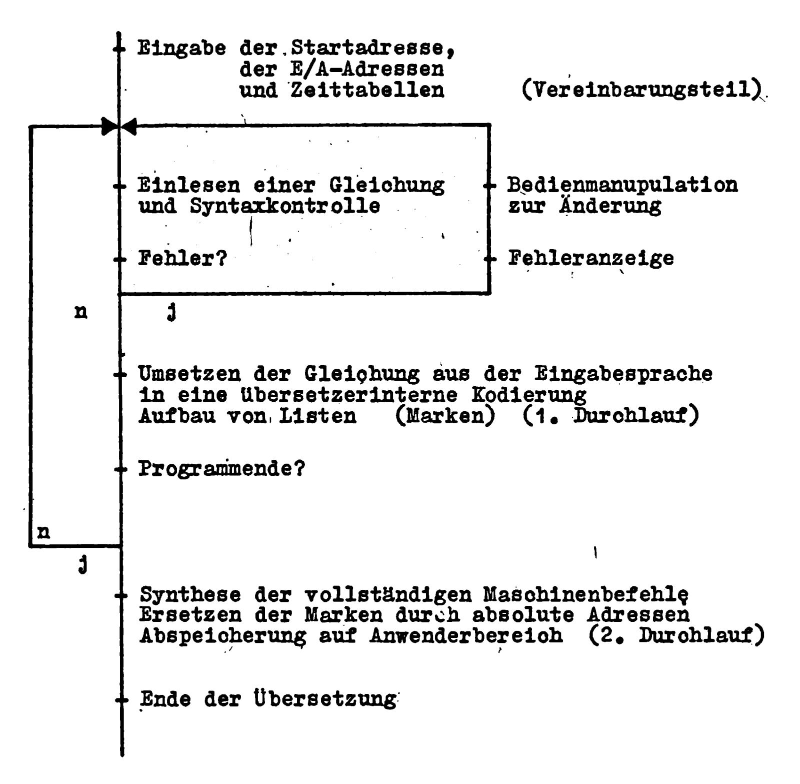


Abb. 10 Grobstruktur eines Übersetzers

Der Übersetzer arbeitet allgemein mit zwei Durchläufen. Im 1. Durchlauf wird zunächst die Anwendermnemonik nach dem Einlesen decodiert und dabei wird festgestellt, ob das Zeichen laut Programmieranleitung überhaupt zulässig ist und ob die Aufeinanderfolge der Zeichen der Vorschrift entspricht.
Diese Maßnahme wird **Syntaxkontrolle** genannt. **Weiterhin wird die** intern abzuarbeitende Befehlsliste erstellt. Für alle Sprunganweisungen werden dabei zunächst symbolische Marken eingesetzt. Die Marken können erst im 2. Durchlauf durch die echten Adressen ersetzt werden, weil dazu die vollständige Befehlsliste im Rechner vorliegen muß.

Für jede der im Punkt 7.2. aufgezeigten problemnahen Quellprogramme können die Verfahren zur Logikabarbeitung aus dem Abschnitt 7.1. genutzt werden. Welche Kombination praktisch zum Einsatz kommt, hängt von vielen Faktoren ab. Außer dem Befehlssatz des Rechners beeinflußt der Verwendungszweck der Steuerung und die vorhandene Bedienperipherie die realisierte Konzeption. Die erreichbaren technischen Daten bei der unmittelbaren Logikverarbeitung weichen nicht wesentlich von den Ergebnissen bei direkter Programmierung des Rechners ab (Tabelle aus 7.1.4.). Durch die starke Formalisierung der Anwendernotierung (Festlegungen in der Programmieranleitung u. U. Programmformular zur Erfassung) und die starke Entlastung von Routinearbeiten entstehen einheitliche Programme, die weniger Fehler enthalten. Da die Ersteller des Quellprogrammes im allgemeinen die rechnerinterne Darstellung nicht kennen, muß für alle Fragen der Fehlersuche und Programmänderung eine Rückübersetzung in die Quellnotierung vom Rechner vorgesehen werden. Dieser Softwareaufwand des Steuerungsherstellers ist beträchtlich. Hier sind die gegenwärtigen Schwierigkeiten einer solchen Gesamtkonzeption zu sehen. Das Programmiergerät ist aufwendig und deshalb von der Steuerung gerätemäßig getrennt. Ohne Programmiergerät kann andererseits der Steuerungsinhalt nicht geändert werden. Eine kostengünstige Variante ergibt sich immer dann, wenn räumlich zentralisiert eine größere Zahl gleichartiger Steuerungen installiert ist. Ein Programmier- und Servicegerät wird in diesem Fall besser ausgelastet.

7.2.2. Interpretierende Arbeitsweise

Diese Arbeitsweise hat einige Gemeinsamkeiten mit der Unterprogrammorganisation, die in der Rechentechnik schon lange genutzt wird. Genau wie dort wird eine gewünschte Funktion symbolisch durch eine Anweisung des Anwenderprogrammes aufgerufen und durch den aktivierten Programmteil praktisch ausgeführt. Danach wird an die nächste Anweisung des Anwenderprogrammes zurückgesprungen. Diese Methode benötigt gegenüber der direkten Programmierung oder der compilierenden Arbeitsweise zusätzliche Organisationszeit zum Umschalten der Programmebene während der zyklischen Gleichungsberechnung. Der Nachteil verliert mit der steigenden Verarbeitungsgeschwindigkeit zukünftiger Prozessoren (Taktfrequenz des U 880 A ist bereits 4 MHz gegenüber den 2,5 MHz der Stan-

dardausführung) an Bedeutung. Der absolute Zeitbetrag je Umschaltung ist stark von der inneren Struktur des Prozessors (Zahl der Register. Adressierungsarten u. ä.) und dem konkreten Befehlsspektrum abhängig. Beim U 880 ergeben sich durch den doppelten Registersatz mit effektiven Austauschbefehlen die vielfältigsten Nutzungsmöglichkeiten der Registerpaare zur zeitsparenden Adreßbehandlung und durch weitere Aspekte nur 20 µs zum Lesen eines Anwenderbefehls im Hauptprogramm einschließlich der Übergabe an den ausgewählten Interpreterteil. Die Befehlsausführungszeit im Interpreterteil weicht nicht wesentlich von den Werten ab, die für die direkte Programmierung ermittelt wurden.

Der ganz entscheidende Vorteil der vorgestellten Arbeitsweise liegt in der einfachen Programmierung einerseits und darin, daß die ursprüngliche Logikstruktur ständig im gespeicherten Anwenderprogramm erhalten bleibt. Dadurch bleiben die Forderungen der Nutzer von PC leicht erfüllbar, daß zur Inbetriebnahme und Fehlersuche das Programm ohne Schwierigkeiten rückgelesen werden kann und der logische Zusammenhang im Schrittbetrieb manuell verfolgbar ist. Logikänderungen können bei geeigneten Speicherelementen unmittelbar an einer gewünschten Stelle des Programms vorgenommen werden, wobei diese Betriebsart viel einfacher als bei compilierender Arbeitsweise realisierbar ist.

Im folgenden wird als Beispiel eine PC auf Interpreterbasis näher vorgestellt. Zunächst sei der Befehlssatz angegeben.

Mnemonik	Hexadezimaler Kode	Erläuterungen
LD	0	Die Abspeicherung der Eingangsvariablen in den RAM (1Wert je Speicherplatz) und die Ausgabe der generierten Steuergrößen wird vorgenommen.
Ju	1	Der Befehlszähler wird mit der im Befehl angegebenen Adresse geladen.
JO	2	Der Befehlszähler wird mit der Adresse geladen, wenn Qu = 0 ist. Bei Nichterfüllung wird der Befehlszähler inkrementriert.
J1	3	Der Befehlszähler wird mit der Adresse geladen, wenn Qo = 1 ist. Bei Nichterfüllung wird der Befehlszähler inkrementiert.

$\bar{x}$	4	Es wird die im Befehl adressierte Variable x ausgewählt und $^{1}Qu:=Qu \wedge \bar{x}$ gebildet.
x	5	Wie $\bar{x}$ nur wird $^{1}Qu:Qu \wedge x$ gebildet
$\bar{z}$	6	Wie $\bar{x}$ nur wird $^{1}Qu:Qu \wedge \bar{z}$ gebildet
z	7	wie $\bar{x}$ nur wird $^{1}Qu:Qu \wedge z$ gebildet
v	8	ODER Verknüpfung Es wird $^{1}Qo:=Qo \vee Qu$ gebildet, danach wird Qu = 1 gesetzt.
(	9	Klammer auf Die Inhalte von Qu und Qo werden gekellert. anschließend wird Qu=1 und Qo=0 gesetzt (Anfangszustand).
)	A	Klammer zu $^{1}Qu:= Qo \vee Qu$ $^{2}Qu:= {^{1}Qu} \wedge Qu_{Keller}$ Anschließend wird Qu und Qo entkellert.
$\bar{=}$	B	Identität Es wird die Indentität von Qu mit dem folgenden Wert (x, $\bar{x}$, z oder $\bar{z}$) gebildet. Das Ergebnis liegt nach Abarbeitung des folgenden Wertes in Qu vor.
$\bar{y}$	C	Speicherung von $\overline{Q_E}$ nach der im Befehl angegebenen Adresse.
y	D	Wie $\bar{y}$, aber mit Speicherung von Q_E
=	E	Zuweisung des Ergebnisses $^{1}Qu:=Qo \vee Qu$ $^{1}Q_E:=Qu$ und Qu=1, Qo=0 gesetzt
w	F	wie y, nur erfolgt die Abspeicherung in den Bereich der Zwischenvariablen.

Der vorgestellte Befehlssatz weicht in seinem Aufbau etwas von der bisherigen Betrachtungsweise ab. In einer Anwendermnemonik ist Operand und Operator nicht mehr zu einer Anweisung vereinigt. Als logische Operatoren sind v; (;);$\bar{=}$; = vereinbart, ein Symbol für "UND" wird nicht angeboten, da dieses auch bei normaler Gleichungsschreibweise wegfällt. Als Operanden sind Eingänge (x), Zwischenvariable (z) und Ausgänge (y) in positiver oder negierter Form vom Anwender notierbar. Dadurch kann fast jede logische Gleichung ohne vorherige Umformung (Sonderfall : Termnegationen) problemlos in eine Anweisungsfolge überführt werden.

Zur Vereinfachung des Interpreters (Zeitersparnis) und des Monitorprogramms (Anzeigeteil) wurde für alle Befehle eine 2-Bytestruktur zugrunde gelegt (Abb. 11).

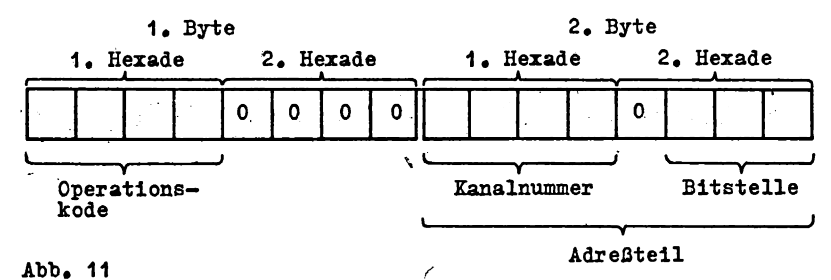


Abb. 11

Bei den adreßlosen Befehlen (Operationen) ist der Inhalt des 2. Bytes beliebig. Für die Anwendernotierung wird damit nicht die maximal mögliche Packungsdichte der Information im Speicher erreicht. Die gute Transparenz des Quellprogramms rechtfertigt diesen partiellen Nachteil. Die interpretative Abarbeitung der Anwendernotierung geschieht je Befehl in 2 Phasen. Abb.12 zeigt den zugehörigen PAP.

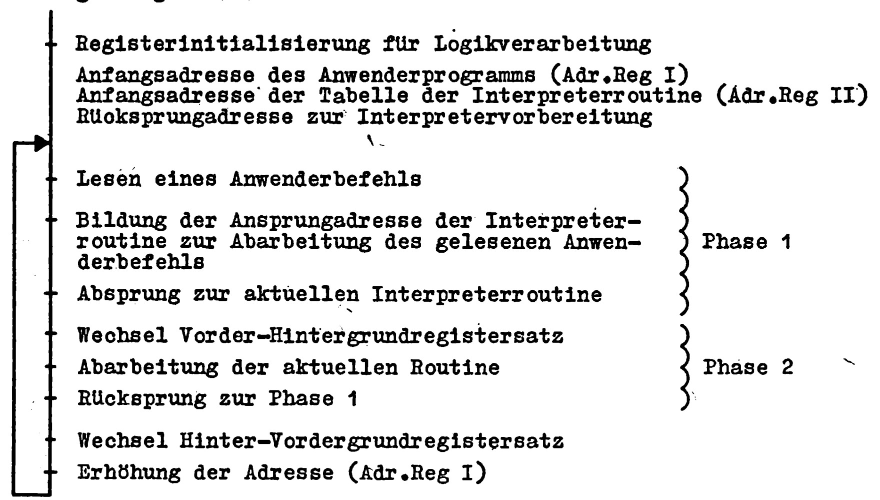


Abb.12 Arbeitsweise des Interpreters

In der 1. Phase wird für jede Anweisung das gleiche Programm-

stück durchlaufen. Es arbeitet mit dem Hauptregistersatz (Vordergrund) und bewirkt das Lesen eines kompletten Anwenderbefehls mit nachfolgender Adreßerhöhung zur Vorbereitung des Lesens der nächsten Anweisung (Adreßregister I).
Für die Phase 2 stehen 16 unterschiedliche Folgen von U 880 Befehlen zur Auswahl. Jede Folge dient der Abarbeitung eines der 16 möglichen Anwenderbefehle. Bei der Erstellung dieser Routine wurde großer Wert darauf gelegt, daß jede Folge ≤ 16 Byte Speicher belegt. Die Anordnung der Routine im Speicher ist folgendermaßen festgelegt. Die 1. Routine zur Abarbeitung des Anwenderbefehls mit dem Operationskode LD ≙ 0 H beginnt auf einem Speicherplatz mit der Adresse XX00 H, d. h. das niederwertige Byte ist identisch Null. Die nachfolgenden Routinen beginnen jeweils im Abstand von 16 Zellen mit dem niederwertigen Adreßteil 10 H, 20 H, ..., F0 H. Im Zusammenhang mit dem konkreten Befehlsaufbau aus Abb. 11 und den beschriebenen Maßnahmen wird die Adreßrechnung zur Ermittlung der Ansprungadresse der jeweils aktivierenden Routine (Bestandteil der Phase 1) trivial, indem das 1. Byte des Anwenderbefehls in den niederwertigen Teil des Adreßregisters II geladen wird. Die stets wiederkehrende und gleich ablaufende Interpretervorbereitung endet mit dem Aufruf der aktuellen Routine. Danach beginnt die Phase 2 mit einem Austauschbefehl (EXX).
Die gesamte Phase 2 arbeitet mit dem Hintergrundregistersatz. Dadurch bleibt der Inhalt des Adreßregisters I und II während dieser Zeit resistent. Zu beachten ist, daß der Akkumulator durch den Befehl EXX nicht beeinflußt wird. Diese Eigenschaft prädestiniert ihn für die Aufgabe der Parameterübergabe aus der Phase 1 an die Phase 2. In der Phase 1 war in den Akku das 2. Byte der Anwendermnemonik eingetragen worden. Es enthält die Operandenadresse bei Operandenbefehlen.

Vom Hintergrundregistersatz bildet das C-Register den Speicher für Konjunktionsergebnisse(Qu)und das D-Register nimmt das Resultat von ODER-Verknüpfungen auf (Qo). Nach Abschluß der Verknüpfungsseite wird durch den Operator "=" das Endergebnis als Steuerwert ins B-Register (Q_E) übernommen und der Grundzustand in Qu und Qo eingestellt. Bei dem Operator "≡" muß ein Merker gesetzt werden (E-Register), um die mit jeder Operandenanweisung verbundene UND-Verknüpfung zu unterdrücken. In Abb. 13 ist für eine Beispielgleichung die Folge der Anwenderbefehle und die jeweilige Belegung der Register angegeben.

Beispielgleichung:

X13 $\overline{\equiv}$ X24vX03 ($\overline{X}$07 Z53 v $\overline{Z}$31) X60vZ15 ≡ $\overline{Y}$02 W02

Befehlsfolge		Registerbelegung U 880 A	B	C 2^1	C 2^0	D 2^1	D 2^0	E
Grundzustand				1	1	0	0	1
X13	50 13	X13			X13			
$\overline{\equiv}$	B0 00							0
X24	50 24	X24			X13$\overline{\equiv}$X24			
v	80 00			1	1		X13≡X24	
X03	50 03	X03			X03			
(	90 00			X03		X13≡X24		
$\overline{X}$07	40 07	$\overline{X}$07			$\overline{X}$07			
Z53	70 53	Z53			$\overline{X}$07 Z53			
v	80 00				1		$\overline{X}$07 Z53	
$\overline{Z}$31	60 31	$\overline{Z}$31			$\overline{Z}$31			
)	A0 00				X03 ( )		X13≡X24	
X60	50 60	X60			X03()X60			
v	80 00				1		[ ]	
Z15	70 15	Z15			Z15			
≡	E0 00		Fkt				[ ] vZ15	
$\overline{Y}$02	C0 02	$\overline{Fkt}$						
W02	F0 02	Fkt						

[ ] ≙ X13 ≡ X24 v X03 (...) X60

Abb. 13 Abarbeitung einer Gleichung

Wie arbeitet der ausführende Teil des Interpreters?
Nicht alle Verfahren der direkten Programmierung (Abschn. 7.1.) eignen sich für die interpretative Arbeitsweise. Das Verfahren bedingter Sprünge ist auf eine problemnahe Quellnotierung nicht anwendbar. Zur Ermittlung der Sprungweite müßte der Interpreter

stets die gesamte Gleichungsnotierung "vorausschauend" kennen. Die PC 600, die mit einem Interpreter auf Sprungbasis arbeitet, benötigt deshalb eine Vorverarbeitung der Anwendernotierung, wodurch aber die ursprüngliche Logikstruktur verschwindet. Für Tabellenverfahren sind Interpreter gut geeignet, aber die Erstellung der Mustertabellen widerspricht den Forderungen problemorientierter Quellnotierung. Es wird deshalb das Verfahren der Gleichungsverarbeitung mittels Logikbefehlen benutzt. Der U 880 arbeitet während der Verarbeitungsphase wie ein spezieller Bitprozessor (siehe Lb 1), wobei die Interpreterroutine die Aufgabe des Steuerwerkes übernimmt. Besondere Aufmerksamkeit muß den Besonderheiten gewidmet werden, die sich aus der byteorientierten Arbeitsweise des universellen Prozessors und der gewünschten Bitmanipulation ergeben (Verschiebung der Bitpositionen, Ausblenden unerwünschter Stellen u. ä.). Ein großer Teil der Schwierigkeiten kann umgangen werden, wenn durch eine Operandenadresse nur ein Bit initialisiert wird. Im Abschn. 7.1. wurden dazu bitorientierte E/A-Karten vorgeschlagen. Diese sind teurer als byteorientierte Peripheriebausteine und in der DDR nicht als Systemkomponenten (ursatron 5000) industriell verfügbar. Für die beschriebene Lösung wird deshalb von einer Byteorganisation der Peripherie ausgegangen. Mit dem PC-Befehl LD wird eine Routine angestoßen, die die Eingangskanäle wortweise in den Akku lädt und danach bitweise in aufeinanderfolgenden Speicherzellen des RAM an die Position 2^0 bringt. In der gleichen Routine werden alle im letzten Verarbeitungszyklus generierten Steuerwerte, die auch bitweise im RAM stehen, durch eine Serien-Parallel-Wandlung zu Byte zusammengeschoben und ausgegeben. Für 64 Eingänge (8 Kanäle) und 32 Ausgänge (4 Kanäle) werden insgesamt $\approx$1 ms benötigt. Würde bei jedem Operandenaufruf ein Einrichten der Bitposition vorgenommen, ergäben sich größere Zeitwerte.

Bezüglich der Klammerschreibweise von Gleichungen bringt die Bytestruktur des Rechners Vorteile. Im C- und D-Register enthält nur die Stelle 2^0 den aktuellen Wert. Mit jeder Öffnung einer Klammer werden die Werte durch Rotation des Inhaltes links (Befehl RLC) um eine Position verschoben (6. Zeile in Abb. 13). Die maximale Klammertiefe ist durch die Wortlänge der Register auf 7 begrenzt.

Als 1. Bepsiel für eine Routine sind nachfolgend die Befehle in Assemblermnemonik für die Abarbeitung der Anweisung "v" (ODER) angegeben:

```
XX80 H   EXX
XX81 H   LD     A, C
XX82 H   AND    1
XX84 H   OR     D
XX85 H   LD     D, A
XX86 H   SET    0, C
XX88 H   JMP    (IX)
```

Unmittelbar nach dem Austauschbefehl wird der Wert der zuletzt programmierten Variable bzw. der bis dahin ausgeführten Konjunktionen aus Qu mit dem Ergebnis bereits abgearbeiteter Disjunktionen disjunktiv verknüpft. Vor dem Absprung in einen weiteren Durchlauf der Interpretervorbereitung wird noch der Grundzustand des UND - Akku (Qu = 1) hergestellt.

Als weiteres Beispiel sei noch die Folge für X gezeigt:

```
XX50 H   EXX
XX51 H   LD     L, A
XX52 H   LD     A, M
XX53 H   LD     B, E
XX54 H   DJNZ   Sprung zur Routine Identität
XX56 H   OR     OFEH
XX58 H   AND    C
XX59 H   LD     A, C
XX5A H   JMP    (IX)
```

Auch in dieser Routine ist in der 1. Zeile der Zusammenhang zwischen dem Kode für die Anweisung X und der Adresse zu erkennen.

Zunächst wird nach dem Registertausch der Wert des adressierten X in den Akku geladen. Mit den 2 eingeschobenen Befehlen wird getestet, ob unmittelbar vor der Programmierung des Operanden der Operator "≡" stand (bei "≡" wird E-Register ungleich 01H gemacht). Der folgende Befehl OR erzwingt für alle Bitpositionen, außer 2^0, den Wert auf 1, so daß mit dem AND nur noch der aktuelle Wert des X den Qu (2^0 des C-Registers) beeinflussen kann. Am Schluß erfolgt wieder der gleiche Rücksprung.

Die angegebenen Folgen der Routine sollen eine Vorstellung über den Aufwand und Zeitbedarf vermitteln. Eine genaue Erklärung des gesamten Interpreters geht weit über den Rahmen des Lehrbriefes hinaus.

Aus den Unterlagen zum U 880 kann man durch Summation der einzelnen Befehlsabarbeitungszeiten für die Routine "v" ≈ 15 µs und für "X" ≈ 22 µs ermitteln. Unter Einbeziehung der stets durchlaufenen Interpretervorbereitung ergibt sich ein durchschnittlicher Wert von 40 µs je Anwenderbefehl. Bei der Abarbeitung von 32 logischen Gleichungen hoher Komplexität wird eine Zykluszeit ≤ 20 ms realisiert.

Für die komplette Steuerung sind weitere Programmteile erforderlich.

Zur Softwarerealisierung von Zeitgliedern wird durch Programmierung des Zeitgeber-Zähler-Bausteins (CTC U 857) ein Interrupt alle 100 ms angemeldet, der aber erst nach der vollständigen Beendigung des laufenden Logikzyklus durch den U 880 Befehl EI (enable interrupt) wirksam wird. In der Interruptroutine zur Behandlung von Zeitgliedern werden die vereinbarten Zwischenvariablen getestet.

Nur, wenn eine getestete Variable 1 ist (entspricht der Erregung eines Zeitrelais), wird die zugehörige Zählerzelle einmal dekrementiert, solange ihr Inhalt ungleich Null ist. Beim Stand Null wird der Wert der Zwischenvariable dem Ausgang zugewiesen.

Auch Zählvorgänge sind durch die IS CTC programmtechnisch realisierbar. Durch ein vereinbartes Anwendersymbol wird eine Routine gestartet, die einen Kanal als Zähler programmiert, wobei der Inhalt im 2. Byte der Anwendermnemonik zur Voreinstellung des Rückwärtszählers genutzt wird. Der Nulldurchgang des Zählerkanals kann über einen Steuerungseingang oder durch Interruptbehandlung auf die Logik zurückwirken.

Eine PC, die den beschriebenen Interpreter nutzt, wurde als Labormuster erprobt /5/. Als Gerätetechnik wurde eine Leiterkarte ZRE 2521 des K 1520 und eine spezielle Leiterkarte mit den Bedien- und Anzeigeelementen eingesetzt. Der gesamte Programmumfang des Interpreters, der Betriebsartenwahl, der Tastaturabfrage mit Kodierung, der Anzeigenansteuerung mit Hexa - 7 - Segmentzuordnung und der Einschaltroutine belegt

1 k Byte PROM. Für die X-, Z- und Y-Variablen, den Stack und die Zeitzellen wird 1 k-Byte RAM benötigt. Da auf der ZRE insgesamt 3 k PROM vorhanden sind, können 2 k Byte mit lauffähigem Anwenderprogramm belegt werden. Für die Programmierung, Testung und Änderung von Anwenderprogrammen muß eine zusätzliche RAM-Karte des K 1520 Systems (K 3520) eingesetzt werden. Wird die 4 k RAM Karte mit CMOS-Elementen genutzt (K 3521), kann bei entsprechender Batteriepufferung das Anwenderprogramm auch im RAM belassen werden.

8. Prozeßinterface

Im Lehrbrief 1 sind Schaltungen für Ein- und Ausgabekarten enthalten. Die prinzipielle Funktionsweise und die Probleme beim industriellen Einsatz sind auch bei Rechnersteuerungen gleichartig vorhanden. Bei Anwendung der zentralen Recheneinheit des Mikrorechners K 1520 (Leiterkartentyp ZRE 2521) müssen bei direktem Anschluß der E/A-Karten an den Bus die elektrischen und mechanischen Bedingungen der Busrichtlinie (TGL 37271/01) erfüllt sein. Dazu gehört beispielsweise, daß jedes Bussignal je Karte mit maximal einem TTL-Eingang belastet werden darf. Weiterhin dürfen ohne aktive Busanforderung einer peripheren Baugruppe und Quittierung durch die CPU (BUSRQ, BAI) Datensignale nur bei Lesebetrieb der CPU (RD) auf den Datenbus (DB0 ... DB7) geschaltet werden. In allen anderen Fällen müssen die Sender der Karten in Busrichtung hochohmig bleiben (tristate). An dieser Stelle sei noch darauf hingewiesen, daß für die Begriffe Lesen (RD) und Schreiben (WR) in Mikrorechnern immer die CPU als Bezugspunkt festgelegt ist. Auch die detaillierte Kontaktbelegung der beiden 58poligen indirekten Steckverbinder ist in der Busrichtlinie enthalten.

Besondere Bedeutung bei umfangreichen Prozeßsteuerungen mit extremen Echtzeitforderungen hat die prozeßbedingte Unterbrechungsmöglichkeit laufender Programme (Interrupt = Signal INT).
In der Schaltkreisfamilie zum U 880 besitzt der Parallel-E/A-Baustein (PIO) die größte Bedeutung für den Betrieb von PC.
In der Betriebsart 3 (Bit-Ein-Ausgabe), die im Rahmen der Einschaltroutine des Rechners programmierbar ist, lassen sich max. 16 Prozeßsignale wahlweise auf L/H oder H/L-Übergang überwachen. Ist beispielsweise für 8 Signale der Ruhezustand (ungefährliche Betriebszustand) low und einer dieser Steuereingänge nimmt pro-

zeßbedingt high-Potential an, so wird ein INT-Signal (Unterbrechung) ausgesandt. Nach der Abarbeitung des gerade laufenden Befehls wird in das entsprechende Programm zur Unterbrechnungsbehandlung (Interruptroutine) gesprungen. Die Reaktionszeit der Rechnersteuerung kann durch diese Möglichkeit für wenige zeitkritische Signale wesentlich gesenkt werden (Richtwert: 1/100 ... 1/10 Zykluszeit). Mit dieser Maßnahme verlängert sich die Reaktionszeit für die zeitunkritischen Steuersignale geringfügig. Für die ordnungsgemäße Arbeitsweise der Steuerung darf der zeitliche Abstand zwischen 2 Interruptsignalen nicht kleiner als die Zykluszeit werden, da die normale Gleichungsberechnung mehrmals hintereinander verhindert würde.

Ein weiteres Problem bei der Kopplung von elektronischen Steuerungen kleinen Signalhubes (TTL- und Rechnersteuerungen) mit industriellen Einrichtungen bildet die räumliche und zeitliche Verschiebung des Bezugspotentials. Durch die teilweise großen Ströme in den Stellgliedern wird in Verbindung mit dem endlichen Leitwert der Verbindungen für das Bezugspotential ein störender Spannungsabfall auftreten.
Zusätzliche Störungen auf dem Bezugspotential entstehen durch kapazitive und induktive Einkopplungen auf das großflächige Netz der Rückleiter. Von den Prozeßrechnerherstellern wird für die Analogsignalverarbeitung der Anschluß von Sensoren über Zweidrahtleitungen an Verstärker mit Differenzeingang empfohlen bzw. gefordert. Durch verdrillte Zweidrahtleitungen kompensiert sich die Auswirkung eines großen Teiles der Störquellen. In speicherprogrammierbaren Steuerungen wird durch galvanische Trennung der Eingänge die Speisung aus räumlich getrennten Quellen (mehrere Netzteile) ermöglicht. Durch Zweidrahtleitungen wird nur der Potentialhub des Nutzsignals am Steuerungseingang wirksam. Eine Verschiebung des Bezugspotentials zwischen Steuerung und Peripherie bleibt ohne Wirkung. Abb. 14 zeigt das Prinzip. In der Praxis hat sich gezeigt, daß bei räumlich konzentrierter Peripherie ein Rückleiter je Stromversorgung ausreichend ist. Trotzdem bleibt bei dieser Methode ein hoher Kostenaufwand, wenn je Eingang ein Optokoppler eingesetzt wird. Bei der PC 600 des VEB Numerik "Karl Marx" wird eine andere Variante der galvanischen Trennung angewendet. Alle Ein- und Ausgangskarten werden zu einer Prozeßeinausgabebesteuerung (PEAS) zusammengefaßt

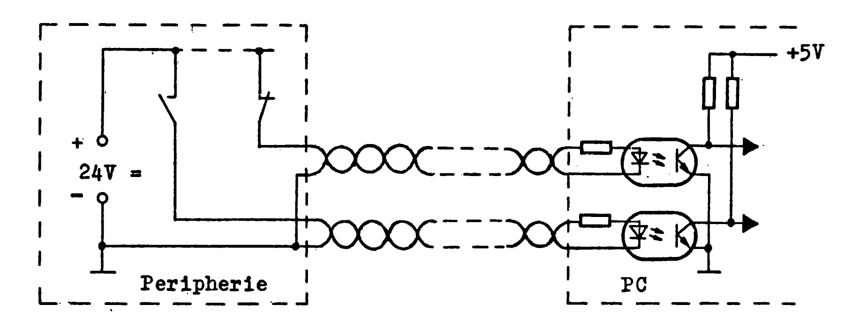


Abb. 14 Prinzip einer störsicheren Peripheriekopplung

und an die Stromversorgungseinheit der Peripherie angeschlossen. Die PEAS ist nicht direkt mit dem Systembus des Rechners verbunden. Alle Anschlüsse der adressierbaren E/A-Karten bilden einen Peripheriebus, der über Optokoppler und PIO auf den Systembus führt. Natürlich müssen Adreß- und Steuersignale auf gleiche Art galvanisch entkoppelt an die PEAS übergeben werden. Bei dieser Konfiguration werden unabhängig von der Zahl der E/A-Signale ≈30 Optokoppler benötigt. Für die Grundausrüstung ergibt sich damit ein relativ großer Aufwand. Mit steigender Zahl der Eingänge wird diese Lösung immer effektiver.

Für zukünftige Lösungen von PC ist mit problemangepaßten IS zu rechnen, die eingangsseitig die Funktionen der Überlastsicherung, Störunterdrückung und galvanischen Trennung realisieren und ausgangsseitig die Signalspeicherung, Leistungsbereitstellung, den Überlastschutz und die Entkopplung übernehmen. Durch solche IS sind bei gleichzeitiger Erhöhung des Integrationsgrades der Mikroprozessoren zu Einchiprechnern sehr kompakte Geräte mit dem vollen Komfort heutiger Systemlösungen zu erwarten.

9. Kurzbeschreibung der industriellen Steuerungen PC 600, ursalog 5010, ursalog 5020

Zum Abschluß der Problematik speicherprogrammierbarer Steuerungen sollen industriell gefertigte Steuereinrichtungen vorgestellt werden. Alle beschriebenen Steuerungen besitzen als Prozessor den universellen wortorientierten Schaltkreis U 880 zur Logikverarbeitung. Ein Vergleich der einzelnen Steuerungen soll nicht Gegenstand der Betrachtungen sein.

9.1 Die speicherprogrammierbare Steuerung PC 600 /8/

Die Steuerung wird neben der im Lehrbrief 1 beschriebenen Steuerung PS 2000 vom VEB Numerik "Karl Marx" Karl-Marx-Stadt gefertigt. Sie ist für die Steuerung räumlich konzentrierter Prozesse mit mittleren Zahlen von Ein- und Ausgängen konzipiert. Das Blockschaltbild zeigt Abb. 15.

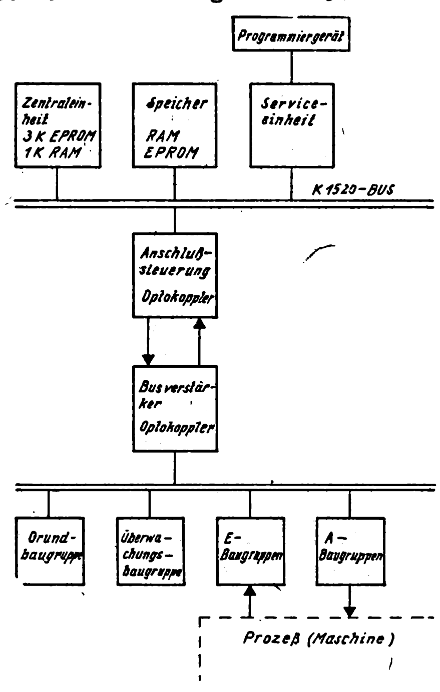


Abb. 15 Blockschaltbild PC 600

Die Steuerung ist funktionell in zwei Bestandteile untergliedert, die durch Optokoppler galvanisch getrennt sind und auch getrennte Stromversorgungsteile besitzen:

- Zentraleinheit (ZE)
- Prozeß-Ein-Ausgabesteuerung (PEAS)

In der Zentraleinheit werden die Logikverarbeitung, der Verkehr mit der PEAS und die Zeitgliedberechnung ausgeführt.
Die PEAS realisiert die Durchschaltung der Eingangsgrößen zur Schnittstelle sowie die Auffächerung der Ausgangssignale auf

die Ausgangsbaugruppen. Je nach Modifikation können maximal 13, 20 oder 40 E/A-Karten angeschlossen werden.

Innerhalb der ZE ist der Systembus des Mikrorechnersystems K 1520 verwendet. Daraus resultiert, daß Kartenbaugruppen des Mikrorechners verwendet werden können. Es wurden die Rechnerkarte K 2521 mit 3K EPROM und 1K RAM verwendet. Weitere Speicherkarten enthalten die Anwenderprogramme (für Inbetriebnahmephase CMOS - RAM) und Diagnoseprogramme. Die Kartenbaugruppen der PEAS enthalten neben den E/A-Baugruppen noch Optokoppler, Busverstärker, Multiplexer, Demultiplexer und Überwachungsglieder für Zeit sowie Diagnosebaugruppen.

Die Programmabarbeitung vollzieht sich in 2 Phasen. In der ersten Phase werden alle Eingangsvariablen eingelesen und im Abbildspeicher der ZE (1K RAM) abgespeichert. Anschließend werden alle im vorherigen Zyklus berechneten Steuerwerte vom Abbildspeicher zu den A-Karten transportiert (max. Dauer 5 ms).
In der zweiten Phase wird das Anwenderprogramm abgearbeitet (max. Dauer 50 ms).

Die Diagnosebaugruppe gestattet eine Diagnose bei laufendem Betrieb; im Fehlerfall kann der Fehlerort angezeigt werden.
Beim Einschalten wird die Betriebsbereitschaft überprüft. Die verfügbaren E/A-Baugruppen sind in folgender Tabelle zu sehen.

Eingabe	16 Bit	24 VGS, 15 mA
Eingabe	8 Bit	110 VWS, 20 mA
Eingabe	32 Bit	12- 24 VGS, 10 mA
Interrupteingabe	16 Bit	12- 24 VGS, 15 mA
A/D-Wandler	8 Analogeingänge	± 5 VGS, 20 kΩ
Ausgabe	8 Bit	über Relais GBR 20.1
Ausgabe	8 Bit	über Geko-Relais
Ausgabe	8 Bit	12- 24 VGS, 2,2 A
Ausgabe	16 Bit	12- 24 VGS, 0,3 A
D/A-Wandler	± 10 V Sollwertausgabe	

Programmierung:
Die Steuerungsaufgabe sollte als logische Gleichung vorliegen. Neben UND, ODER, NEGATION sind Klammern beliebiger Schachtelung zugelassen. Jede Gleichung muß mit einer Satznummer beginnen. Daran schließt sich die Ergebnisseite der Gleichung an. Nach dem "=" folgt der Bool'sche Ausdruck (Verknüpfungsseite).

Auf der Verknüpfungsseite können Eingänge, Ausgänge, Zwischenwerte, Zeitgliedeingänge, -ausgänge, Multiplexer- und A/D-Wandlereingänge geschrieben werden.

Auf der Ergebnisseite dürfen Ausgänge, Zwischenwerte, Zeitgliedeingänge und Fehlersignale stehen.

Jeder Wertetyp ist durch einen Buchstaben gekennzeichnet, dem die Kanal- und Bitadresse nachgestellt ist.

Das aufgestellte Programm wird über ein gesondertes Programmiergerät eingeschrieben und in eine für die interpretative Arbeitsweise geeignete Form gebracht. Vom Steuerungshersteller werden Arithmetikunterprogramme für die Funktionen Addition, Subtraktion, Multiplikation, Division, Vergleich, Umkodierung und Zählen für ganze Zahlen geliefert.

9.2. Speicherprogrammierbare Steuereinrichtung ursalog 5010 /6/

Die speicherprogrammierbare Steuerung ursalog 5010 wird im VEB Elektro-Apparate-Werke Berlin gefertigt.
Sie bereichert das Bausteinsystem ursalog 4000, welches für verbindungsprogrammierte elektronische Steuerungen konzipiert ist, wesentlich und besitzt die gleichen Anschlußparameter. Dadurch wird die Nutzung der verschiedenen E/A-Baugruppen und Stromversorgungseinheiten möglich.

Die Steuerung ist in einem Karteneinschub 215 x 170 x 40 mm untergebracht. Sie besitzt 32 störfeste binäre Eingänge und 16 kurzschlußfeste Ausgänge im Systempegel.
Dies bedeutet, daß jeder Ausgang nur mit 3 mA belastet werden darf. Die Nachschaltung von Ausgabekarten des Systems ursalog 4000 ist unerläßlich. Das Betriebssystem umfaßt die Ein- und Ausgabe, den Aufruf der Anwenderprogramme und Testroutinen.

Zur Realisierung des Steueralgorithmus stehen die Bausteine UND, ODER, EXCLUSIVES ODER, ZEITGLIED, ZÄHLER, SPRUNG und TRANSPORT zur Verfügung. Die prinzipielle Vorgehensweise bei der Programmierung ist in Abschn. 7.2. gezeigt.

Für die Erstellung der Anwendersteuerprogramme steht das Programmiergerät PG ursatron 5000 zur Verfügung.
Hier wird das Quellenprogramm in das Objektprogramm übersetzt. Die Eingabe kann über Tastatur oder Datenträger erfolgen. Das

Objektprogramm (Maschinenkode) darf 2k Byte nicht überschreiten. Eine Relation zum Umfang des Quellenprogramms wird vom Hersteller nicht angegeben.
Bei der Inbetriebnahme muß die Steuerung durch eine Inbetriebnahmeverarbeitungseinheit ersetzt werden. Das Anwenderprogramm ist auf einem RAM-Speicher abgelegt und kann über das anzuschließende Programmiergerät beeinflußt werden. Das fehlerfreie Programm wird in einem EPROM abgelegt, der in den Karteneinschub ursalog 5010 gesteckt wird.
Zur Funktionskontrolle der Steuerung wird das Baugruppenprüfgerät ursalog 4000 verwendet.
Die Steuerung wird auf dieses aufgesteckt und mit Testprogrammfolgen geprüft.

9.3. Speicherprogrammierbare Steuerung ursalog 5020 /7/

Die Steuerung ursalog 5020 wird ebenfalls im VEB Elektro-Apparate-Werke Berlin gefertigt. Sie dient zur digitalen Informationsverarbeitung auch dezentraler Prozesse bei mittleren und großen Zahlen der Ein- und Ausgänge.
Abb. 16 zeigt das Blockschaltbild der Steuerung.

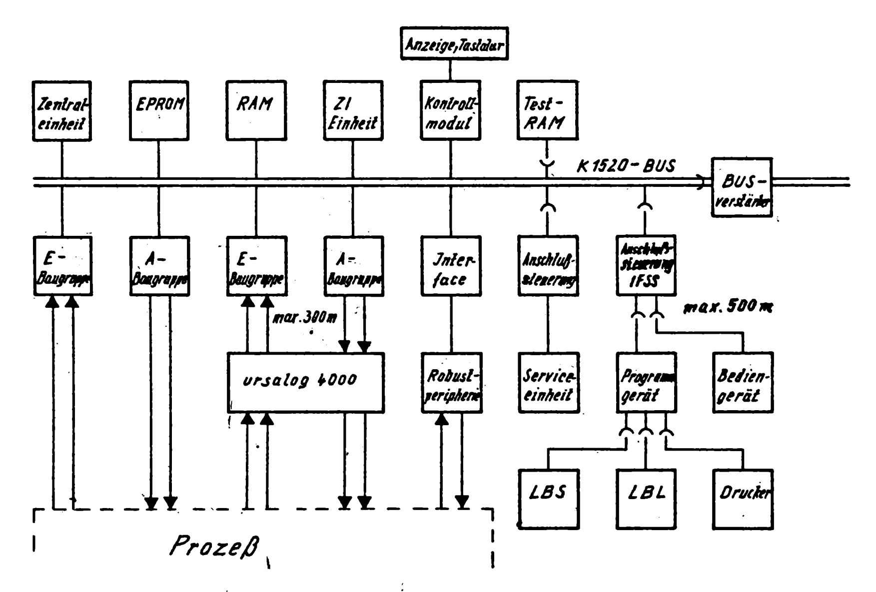


Abb. 16 Blockschaltbild ursalog 5020

Die einzelnen Baugruppen sind über den Mikrorechner-Bus des Systems K 1520 miteinander verbunden. Die in der Grundeinheit eingesetzten Karteneinschübe (KES) sind Teile der Systeme K 1520, ursalog 5020 und ursadat 5000. Über die Busverstärkereinheit kann ein zweiter Einbaurahmen genutzt werden, in dem aber nur Peripherie-KES enthalten sein dürfen.

Der Kontrollmodul mit Tastatur-Anzeige-Vorsatz sollte in jeder Grundeinheit vorhanden sein, da er neben einer Testung des Anwenderprogramms zur Fehlersignalisation genutzt wird.

Zur Eingabe der Prozeßvariablen und Ausgabe der Steuerwerte bestehen 3 Möglichkeiten:

- Einsatz von Baugruppen, die direkt mit der Zentraleinheit zusammenarbeiten. Dazu gehören u. a.:

Digitaleingabe	16 Bit	5, 12, 24 VGS	optoelektronische Trennung
Digitaleingabe	128 Bit	mit Multiplexer	
Digitaleingabe	32 Bit	zur Ankopplung	ursalog 4000
Digitalausgabe	24 Bit	max. 60 VGS,	0,5 A (Relais)
Digitalausgabe	32 Bit	zur Ankopplung	ursalog 4000
Digitalausgabe	32 Bit	max. 60 VGS,	0,12 A (Transistor)

- Einsatz von Baugruppen des Systems ursalog 4000 über o. g. E/A-Karten.

- Einsatz von Baugruppen der Robustperipherie, die über Interfacebaugruppen angeschlossen werden müssen. Dazu gehören u. a.:

Kontakteingangsbaugruppe	32 Bit	24 VGS	10 mA
Digitaleingangsbaugruppe	32 Bit	ursalog 4000	
Kontaktausgabebaugruppe	8 Bit	30 VGS	3 A
Kontaktausgabebaugruppe	8 Bit	220 V WS	15 VA

Eine Interfacebaugruppe (IFSS) ist für den Anschluß eines Programmiergerätes sowie des Bediengerätes vorgesehen.
Die gesamte Steuerung ist modular aufrüstbar. Als Grenzen können genannt werden:

- Zahl der Peripherieadressen: 256
  Eine Peripherie KES belegt, wie auch Interfacebaugruppen, Kontrollmodul und Zentraleinheit, 8 Adressen.
- Speicherkapazität des Abbildspeichers 2K Bit.
  Für Eingänge, Ausgänge und Zwischenvariable wird jeweils 1 Bit belegt. Zeit- und Zählglieder belegen 8 Bit.

Programmierung:

Die Steuerungsaufgabe wird in der problemorientierten Sprache PROLOG 2 formuliert (s. auch Abb. 8).
Mittels Tastatur und Datenträger wird es in das Programmiergerät eingelesen. Ein Kontrollesen auf dem Monitor ist möglich.
Mit dem Programmiergerät sind u. a. folgende Funktionen ausführbar:
- Übersetzung des Quellenprogramms in das Objektprogramm mit Syntaxkontrolle und eventueller Ausgabe auf Datenträger,
- Programmkorrekturen über Tastatur,
- Beschreiben des Test-RAM zur Inbetriebnahme,
- Programmierung der EPROMs,
- Rücklesen des Anwenderprogramms (Maschinenkode) und Anzeige in Quellenprogrammdarstellung.

Zur Realisierung von Bedienfunktionen wie Eingriffe in die Programmabarbeitung, Parameteränderung .... wird ein Bediengerät angeboten. Es wird über eine Interfacebaugruppe angeschlossen. Eine Serviceeinheit dient als Hilfsmittel bei der Inbetriebnahme und bei der Fehlersuche beim Service der Grundeinheit.

Literaturverzeichnis

/1/ Schwarz, W./Meyer, G./ Eckhardt, D. — Mikrorechner. Verlag Technik, Berlin 1980

/2/ Weller, W./Wilke, H. — Programmierbare Steuereinrichtungen. Verlag Technik, Berlin 1981. Reihe Automatisierungstechnik

/3/ Programmierhandbuch für SPS ursalog 5020. Institut für Regelungstechnik, Fachabteilung Steuerungstechnik, 10/1981

/4/ Programmieranleitung PC 601 ... PC 603. VEB Numerik "Karl Marx" Karl-Marx-Stadt, 1981

/5/ Fischer, D./Mauersberger, K./ Wächter, R. — Speicherprogrammierbare Steuerung. Technische Hochschule Karl-Marx-Stadt, NV 79/82

/6/ Kurzinformation ursalog 5010. VEB EAW Berlin

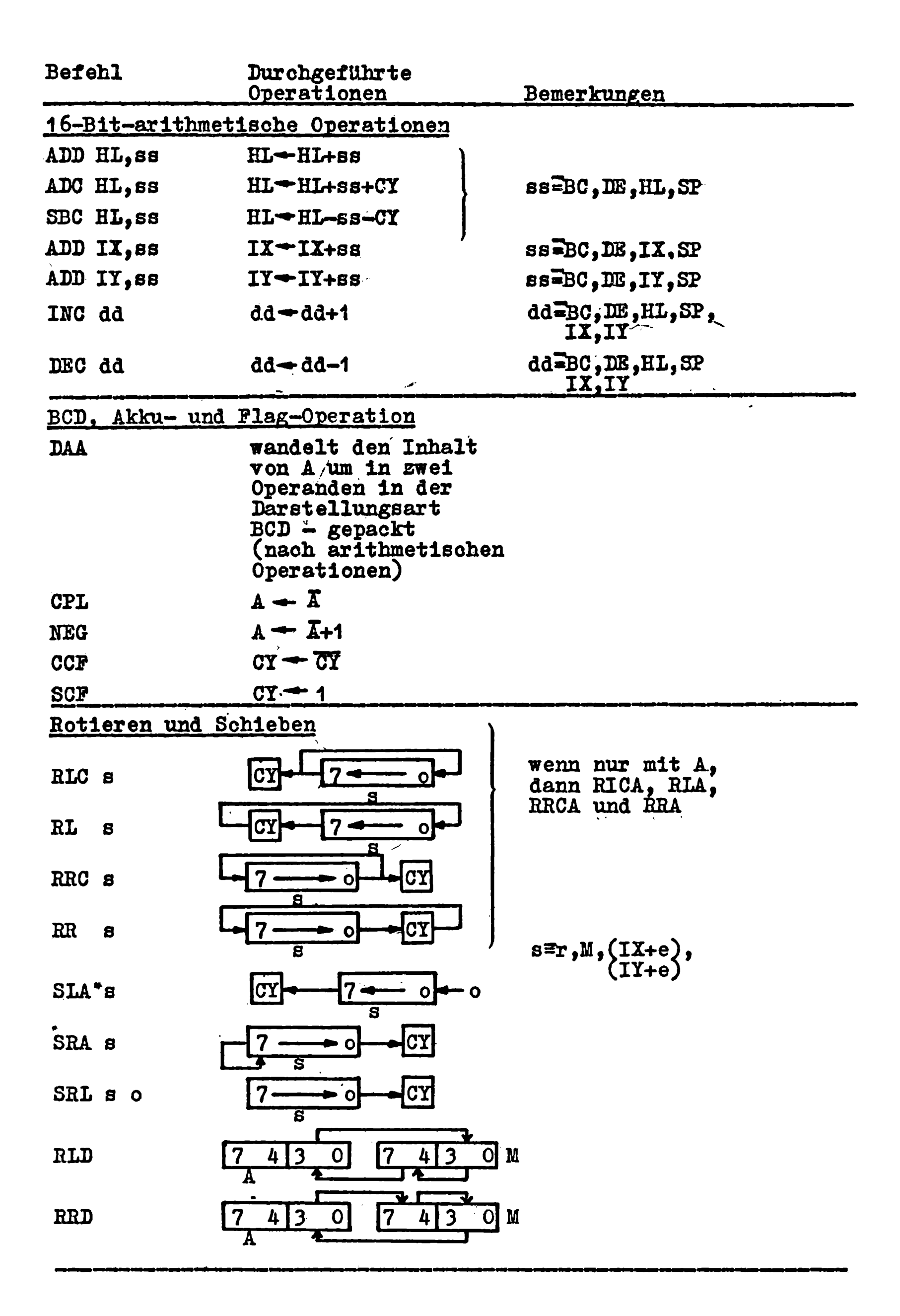
/7/ Kundeninformation ursalog 5020. VEB EAW Berlin

/8/ Funktionsbeschreibung PC 601 ... 603. VEB Numerik "Karl Marx", Karl-Marx-Stadt, 1981

Anhang Befehlssatz des U 880

Befehl	Durchgeführte Operation	Bemerkungen
8-Bit-Ladebefehle		
LD r_1,s	$r_1 \leftarrow s$	s=r_2,n,M,(IX+e), (IY+e)
LD d,r	$d \leftarrow r$	d=M,r,(IX+e), (IY+e)
LD d,n	$d \leftarrow n$	d=M,(IX+e), (IY+e)
LD A,s	$A \leftarrow s$	s=(BC),(DE),(nn), I,R
LD d,A	$d \leftarrow A$	d=(BC),(DE),(nn), I,R
16-Bit-Ladebefehle		
LD dd,nn	$dd \leftarrow nn$	dd=BC,DE,HL,SP, IX,IY
LD dd,(nn)	$dd \leftarrow (nn)$	dd=BC,DE,HL,SP, IX,IY
LD(nn),ss	$(nn) \leftarrow ss$	ss=BC,DE,HL,SP, IX,IY
PUSH ss	$(SP-1) \leftarrow ss_H$; $(SP-2) \leftarrow ss_L$	ss=BC,DE,HL,AF, IX,IY
LD SP,ss	$SP \leftarrow ss$	ss=HL,IX,IY
POP dd	$dd_L \leftarrow (SP)$; $dd_H \leftarrow (SP+1)$	dd=BC,DE,HL,AF, IX,IY
Registeraustausch		
EX,DE,HL	$DE \leftrightarrow HL$	
EXAF	$AF \leftrightarrow AF'$	
EXX	$\begin{pmatrix} BC \\ DE \\ HL \end{pmatrix} \leftrightarrow \begin{pmatrix} BC' \\ DE' \\ HL' \end{pmatrix}$	
EX (SP),ss	$(SP) \leftrightarrow ss_L$; $(SP+1) \rightarrow ss_H$	ss=HL,IX,IY

Befehl	Durchgeführte Operation	Bemerkungen
Block-Transfers		
LDI	(DE) ← M, DE ← DE+1 HL ← HL+1, BC ← BC-1	
LDIR	(DE) ← M, DE ← DE+1 HL ← HL+1, BC ← BC-1 wiederholend bis BC=Ø	
LDD	(DE) ← M, DE ← DE-1 HL ← HL-1, BC ← BC-1	
LDDR	(DE) ← M, DE ← DE-1 HL ← HL-1, BC ← BC-1 wiederholend bis BC=Ø	
Blocksuchbefehle		
CPI	A-M, HL ← HL+1 BC ← BC-1	
CPIR	A-M, HL ← HL+1 BC ← BC-1, wiederholend bis BC=Ø oder A=M	A-M setzt nur die Flags. A wird nicht verändert.
CPD	A-M, HL ← HL-1 BC ← BC-1	
CPDR	A-M, HL ← HL-1 BC ← BC-1, wiederholend bis BC=Ø oder A=M	
8-Bit-arithmetische und logische Operationen		
ADD s	A ← A+s	
ADC s	A ← A+s+CY	CY ist das Überlauf-Flag (Carry-Flag)
SUB s	A ← A-s-CY	
SBC s	A ← A-s-CY	s=r,n,M,(IX+e), (IY+e)
AND s	A ← A ∧ s	
OR s	A ← A ∨ s	
XOR s	A ← A ⊕ s	
CMP s	A-s	s=r,n,M
INC d	d ← d+1	d=r,M,(IX+e), (IY+e)
DEC d	d ← d-1	



Befehl	Durchgeführte Operationen	Bemerkungen
16-Bit-arithmetische Operationen		
ADD HL,ss	HL←HL+ss	ss=BC,DE,HL,SP
ADC HL,ss	HL←HL+ss+CY	ss=BC,DE,HL,SP
SBC HL,ss	HL←HL–ss–CY	ss=BC,DE,HL,SP
ADD IX,ss	IX←IX+ss	ss=BC,DE,IX,SP
ADD IY,ss	IY←IY+ss	ss=BC,DE,IY,SP
INC dd	dd←dd+1	dd=BC,DE,HL,SP, IX,IY
DEC dd	dd←dd–1	dd=BC,DE,HL,SP IX,IY
BCD, Akku- und Flag-Operation		
DAA	wandelt den Inhalt von A um in zwei Operanden in der Darstellungsart BCD – gepackt (nach arithmetischen Operationen)	
CPL	A ← $\bar{A}$	
NEG	A ← $\bar{A}$+1	
CCF	CY ← $\overline{CY}$	
SCF	CY ← 1	
Rotieren und Schieben		
RLC s		wenn nur mit A, dann RLCA, RLA, RRCA und RRA
RL s		
RRC s		
RR s		
SLA s		s=r,M,(IX+e),(IY+e)
SRA s		
SRL s		
RLD		
RRD		

Befehl	Durchgeführte Operationen	Bemerkungen
Bit Setzen, Rücksetzen, Testen		
BIT b,s	$Z \leftarrow \bar{s}_b$	Z ist das Null-Flag (Zero-Flag)
SET b,s	$s_b \leftarrow 1$	s=r,M,(IX+e),(IY+e)
RES b,s	$s_b \leftarrow \emptyset$	
Ein-/Ausgabe		
IN n	$A \leftarrow (n)$	
IN r	$r \leftarrow (C)$	Flags werden verändert
INF	$F \leftarrow (C)$	
INI	$M \leftarrow (C), HL \leftarrow HL+1$ $B \leftarrow B-1$	
INIR	$M \leftarrow (C), HL \leftarrow HL+1$ $B \leftarrow B-1$ wiederholend bis $B=\emptyset$	
IND	$M \leftarrow (C), HL \leftarrow HL-1$ $B \leftarrow B-1$	
INDR	$M \leftarrow (C), HL \leftarrow HL-1$ $B \leftarrow B-1$ wiederholend bis $B=\emptyset$	
OUT n	$(n) \leftarrow A$	
OUT r	$(C) \leftarrow r$	
OUT I	$(C) \leftarrow M, HL \leftarrow HL+1$ $B \leftarrow B-1$	
OUTIR	$(C) \leftarrow M, HL \leftarrow HL+1$ $B \leftarrow B-1$ wiederholend bis $B=\emptyset$	
OUTD	$(C) \leftarrow M, HL \leftarrow HL-1$ $B \leftarrow B-1$	
OUTDR	$(C) \leftarrow M, HL \leftarrow HL-1$ $B \leftarrow B-1$ wiederholend bis $B=\emptyset$	

Befehl	Durchgeführte Operationen	Bemerkungen
Sprungbefehle		
JMP nn	PC ← nn	cc { NZ;PO / Z ;PE / NC;P / C ;M
Jcc nn	Wenn die Bedingung zutr. PC←nn, sonst weiter	
JR e	PC←PC+e	
JRkk e	Wenn die Bedingung zutrifft PC←PC+e, sonst weiter	kk { NZ;NC / Z ;C
JMP(SS)	PC← ss	ss=HL,IX,IY
DJNZ e	B ← B-1, wenn B=∅ weiter, sonst PC ← PC+e	
Unterprogrammaufrufe		
CALL nn	(SP-1) ← PC_H (SP-2) ← PC_L, PC← nn,SP←SP-2	
CAcc nn	Wenn die Bedingung zutrifft Operation wie CALL nn, sonst weiter	cc { NZ;PO / Z;PE / NC;P / C;M
Restarts		
RST p	(SP-1) ← PC_H (SP-2) ← PC_L, PC_H ← ∅ PC_L ← p,SP ← SP-2	p=∅H,8H,18H, 2∅H,28H,3∅H, und 38H
Rücksprünge		
RET	PC_L ← (SP) PC_H ← (SP+1),SP ← SP+2	
Rcc	Wenn die Bedingung nicht zutr. weiter, sonst Operation wie bei RET	cc { NZ;PO / Z;PE / NC;P / C;M
RETI	Rückkehr von Interrupt Operation wie bei RET	
RETN	Rückkehr von nichtmaskierbaren Interrupt	

Verschiedenes

NOP	Leerbefehl (keine Operation)	
HALT	CPU geht in Halt	
DI	Interrupts sperren	
EI	Interrupts zulassen	
IMØ	Interrupt-Mode Ø setzen	Ruf über eingespeiste Adresse
IM1	Interrupt-Mode 1 setzen	Ruf über ØØ38H
IM2	Interrupt-Mode 2 setzen	Ruf über Vektoradresse

Zeichenerklärung:

b: Bit-Position in einem Register oder einer Speicherstelle

cc: Statusbedingungscode
Erlaubte Bedingungen: NZ: ungleich Null
Z: gleich Null
NC: kein Übertrag
C: Übertrag
PO: ungerader bzw. kein Überlauf
PE: gerader bzw. Überlauf
P: positiv
N: negativ

d: 8-Bit-Zielregister

dd: 16-Bit-Zielregister oder Zieladresse im Speicher

e: 8-Bit-vorzeichenbehaftetes-Zweierkomplement der Distanz bei relativen Sprüngen oder indizierter Adressierung

p: 8 spezielle Zieladressen im Speicherbereich 00H bis FFH (dezimal 0, 8, 16, 24, 32, 40, 48 und 56)

n: 8-Bit-Binärzahl

nn: 16-Bit-Binärzahl

r: allgemeines 8-Bit-Register (A,B,C,D,E,H oder L)

s: 8-Bit-Senderegister oder Speicherstelle

s_b: Bit in einem bestimmten 8-Bit-Register oder Speicherstelle

ss: 16-Bit-Senderegister der Speicherstelle

Index "L": niederwertige 8-Bit eines 16-Bit-Registers

Index "H": höherwertige 8-Bit eines 16-Bit-Registers

( ) Zeichen zwischen den Klammern stellen einen Zeiger auf eine Speicherstelle oder ein E/A-Port dar.

M, (HL) durch den Inhalt von Register H und L adressierte Speicherstelle. M und (HL) sind identisch.